CSC-EPL-90/004

(2)

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT

Wang Laboratories, Incorporated

SVS/OS CAP 1.0

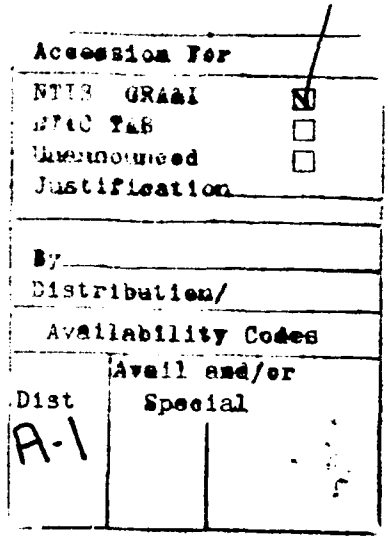
28 September 1990

Approved for Public Release:
Distribution Unlimited

FINAL EVALUATION REPORT

Wang Laboratories, Inc.

SVS/OS CAP 1.0



NATIONAL

COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

September 28, 1990

Report No. CSC-EPL-90/004
Library No. S236,000

FOREWORD

This publication, the *Final Evaluation Report, Wang Laboratories, Inc., SVS/OS CAP 1.0*, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center." The purpose of this report is to document the results of the formal evaluation of Wang SVS/OS CAP 1.0. The requirements stated in this report are taken from *Department of Defense Trusted Computer System Evaluation Criteria*, dated 26 December 1985.

Approved:

Patrick R. Gallagher, Jr.
Director,
National Computer Security Center

September 28, 1990

ACKNOWLEDGEMENTS

Team Members

Timothy J. Bergendahl
Duane A. Souder

The MITRE Corporation
Bedford, MA

Anthony J. Apted
James L. Arnold Jr.
Ronald J. Bottomly
R. Kris Britton

National Security Agency
Trusted Products and Network
Security Evaluation Division
Fort George G. Meade, MD

Technical support was also provided by Deborah Downs, Virgil Gligor, Paul Hager, Karina Hasler, Kim McNicholas, Kathleen M. Tessier, Grant Wagner, and Allen Wentink.

For their contributions to this document, acknowledgement is given to Frank Belvin, Richard B. Newton, and David W. Summers.

Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

The security protection provided by Wang Laboratories, Inc., Secure Virtual Storage (VS) Operating System with Controlled Access Protection (CAP), Version 1.0, SVS/OS CAP 1.0, configured according to the most secure manner described in the associated Trusted Facility Manual, running on the Wang VS Product Family (as described in Appendix A), has been examined by the National Security Agency (NSA). The security features of SVS/OS CAP 1.0 were examined against the requirements specified by the *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), dated 26 December 1985.

The NSA evaluation team has determined that the highest class at which SVS/OS CAP 1.0 satisfies all the specified requirements of the TCSEC is class C2.

A system that has been rated as being a class C2 system provides a trusted computing base (TCB) that enforces discretionary access control protection and, through the inclusion of audit capabilities, accountability of subjects for actions they initiate.

SVS/OS CAP 1.0 runs on the Wang VS Product Family, a series of 32-bit super-minicomputers that can support from 512 KB to 32 MB of addressable physical storage. SVS/OS CAP 1.0 consists of the VS/OS operating system, Release 7.33, Extended Security Access Controls (ESAC), and I/O Device Management features.

SVS/OS CAP 1.0 is a user-friendly, menu-driven, general-purpose, time-sharing system which supports identification and authentication of users, discretionary access control, object reuse, and auditing.

Wang has a configuration management plan in effect for this product for future Ratings Maintenance Phase (RAMP) participation.

Chapter 1

Introduction

In August 1989 the National Security Agency (NSA), Trusted Products and Network Security Evaluation Division, began a formal product evaluation of SVS/OS CAP 1.0, a product of Wang Laboratories, Inc.. The objective of this evaluation was to rate SVS/OS CAP 1.0 against the *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC) [10], and to place it on the Evaluated Products List (EPL) with a final rating. This report documents the results of that evaluation.

Material for this report was gathered by the NSA formal evaluation team through documentation, interaction with system developers, and experience using Wang Laboratories, Inc., systems.

1.1 Evaluation Process Overview

The Department of Defense Computer Security Center was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985 the name of the organization was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the DoD Trusted Computer System Evaluation Criteria (TCSEC) was written. The TCSEC establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The TCSEC levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the TCSEC by an NSA, Trusted Product and Network Security evaluation team.

The NSA supports the creation of secure computer products in varying stages of development from initial design to those that are commercially available. Preliminary to an evaluation, products must go through the Proposal Review Phase. This phase includes an assessment of the vendor's capability to create a secure system and complete the evaluation process. To support this assessment, a Preliminary Technical Review (PTR) of the system is done by the NSA. This consists of a quick review of the current state of the system by a small, but expert, team and the creation of a short report on the state of the system. If a vendor passes the Proposal Review Phase they will enter a support phase preliminary to evaluation. This support phase has two steps, the Vendor Assistance Phase (VAP) and the Design Analysis Phase (DAP). During VAP, the newly assigned team reviews design specifications and answers technical questions that the vendor may have about the ability of the design to meet the requirements. A product will stay in VAP until the vendor's design, design documentation, and other required evidence for the target TCSEC class are complete and the vendor is well into implementation. At that time, the support moves into DAP.

The primary thrust of DAP is an in-depth examination of a manufacturer's design for either a new trusted product or for security enhancements to an existing product. DAP is based on design documentation and information supplied by the industry source, it involves little "hands on" use of the system, but during this phase the vendor should virtually complete implementation of the product. DAP results in the production

of an Initial Product Assessment Report (IPAR) by the NSA assessment team. The IPAR documents the team s understanding of the system based on the information presented by the vendor. Because the IPAR contains proprietary information and represents only a preliminary analysis by the NSA, distribution is restricted to the vendor and the NSA.

Products that have completed the support phase with the successful creation of the IPAR, enter formal evaluation. Products entering formal evaluation must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal evaluation is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the TCSEC. The analysis performed during the formal evaluation requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal evaluation results in the production of a final report and an Evaluated Products List entry. The final report is a summary of the evaluation and includes the EPL rating which indicates the final class at which the product satisfies all TCSEC requirements in terms of both features and assurances. The final report and EPL entry are made public.

After completion of the Formal evaluation phase, products rated at B1 and below enter the rating maintenance phase (RAMP). The rating maintenance phase provides a mechanism to entend the previous rating to a new version of an evaluated computer system product. As enhancements are made to the computer product the ratings maintenance phase ensures that the level of trust is not degraded.

Rating Maintenance is accomplished by using qualified vendor personnel to manage the change process of the rated product during the maintenance cycle. These qualified vendor personnel must have strong technical knowledge of computer security and of their computer product. These trained personnel will oversee the vendor's computer product modification process. They will demonstrate to the Trusted Product and Network Security Evaluation Division that any modification or enhancements applied to the product preserve the security mechanisms and maintain the assurances required by the TCSEC for the rating previously awarded to the evaluated product.

1.2 Background and History

Wang Laboratories, Inc., which is headquartered in Lowell, Massachusetts, was founded in 1951 in Boston, Massachusetts by Dr. An Wang. The company was initially involved with developing word processing tools for use on small computer systems, and became a leader in this field. Today Wang manufactures and sells a wide range of computer systems.

In October 1978 the first VS systems were introduced. The VS60 and the VS80 belonged to the 2200 VS family. This family was based on a 32-bit machine architecture using a 16-bit bus. Word processing was first integrated with data processing in the VS80. In June 1979 the first 32-bit system, the VS100, was introduced.

Today, the product line is called the Wang VS Product Family. All Wang VS systems are based on a 32-bit machine architecture, and all Wang VS systems implement the same VS/OS operating system.

Since its inception, Wang VS/OS has provided identification and authentication of users and access controls on files, the latter via *file protection classes*. With the Release 7 Series architectural changes were made to the operating system to provide a basis for a more secure system. Some of these changes include improved auditing of security relevant events, password enhancements, object reuse, and the implementation of access

control lists.

1.3 Document Organization

This report consists of five chapters and four appendices. Chapter 1 is an introduction. Chapter 2 is a system overview and provides information about SVS/OS hardware and software architecture, including protection mechanisms. Chapter 3 includes a discussion of the *Ratings Maintenance Phase* (RAMP) pertaining to SVS/OS. Chapter 4 presents a mapping of SVS/OS features to the C2 requirements in the TCSEC. Chapter 5 contains evaluator's comments. The four appendices identify the evaluated hardware components, the evaluated software components, provide a glossary, and present a bibliography.

1.4 Conventions

For consistency with Wang's conventions, the term "workstation" is used rather than "terminal" throughout this report. Also, each time a Wang-specific term is defined, it is *emphasized* only where it is defined and used normally elsewhere.

A number of style conventions have also been adopted which help the reader to identify various entities described in text. The following is a table of entity types followed by their associated text style.

Entity type	**Text style**
Register	**Bold face**
Central Processor (CP) Instruction	**Bold face**
(Device) Function	**Bold face**
CP type	*Italics*
Supervisor call	`Typewriter style`
Trusted Process	`Typewriter style`
Filename	`Typewriter style`
Keyboard Key	`Typewriter style`
Port or Device numbers	`Typewriter style`

The use of the notation SVS/OS within this document implies the evaluated product SVS/OS CAP 1.0.

Chapter 2

System Overview

2.1 Hardware Architecture

The evaluated system is comprised of SVS/OS CAP 1.0 running on a base Central Processor (CP) with a collection of attached controllers and peripheral devices (see "Evaluated Hardware Components," Appendix A, page 66). The base CP can be one of several CP-types (e.g., *CP8*). The CP-types vary at a low level (i.e., in implementation), but appear identical at the interface used by SVS/OS. This is accomplished through specialized (i.e., CP-dependent) microcode which, although implemented differently among the CP-types, provides the same functions (e.g., instruction set, ring structure, trap handling) known as the Machine Language Architecture (MLA).

The relevant differences between the CP-types (including memory, controllers, and devices) are discussed in this section.

2.1.1 The Central Complex

CPs, memory, and I/O Controllers (IOCs)[1] will be discussed in this section. This discussion will present an architectural overview of the basic hardware components with emphasis on the generic hardware architecture. Differences in hardware components will be covered when such differences are seen at the software level, or are otherwise considered to be relevant. At the end of this section, a description of hardware system initialization is provided to tie the previously covered concepts together.

The VS hardware family provides a general purpose architecture packaged as 32-bit super-minicomputers. These machines support virtual memory, distinct process levels (a ring architecture), and implement a microcoded instruction set with logical, arithmetic, queue manipulation, semaphore manipulation, and stack manipulation instructions. Main memory, which consists of 32-bit words, and which is divided into 2 KB pages, is implemented as semiconductor Random Access Memory (RAM) with either on-board automatic Error Correction Circuitry (ECC) (*CP4*, *CP7*, *CP8*, *CP10*, and *CP12*), or parity checking (*CP9*), and up to 32 MB of storage capacity. All I/O operations are controlled by IOCs.

CP Architecture

Wang provides the following CP types (see "Description of Specific CPs," page 9): *CP4*, *CP7*, *CP8*, *CP9*, *CP10*, and *CP12*. The CP provides mechanisms for controlling and accessing main memory; decimal, integer, floating point, and decimal floating point operations; basic logic operations; and flow-control/sequencing operations. It is also responsible for initiating communications between memory and external devices (see

[1] IOC is a general term refering to a collection of devices (i.e., "I/O Controllers," "I/O Processors," and "Bus Processors"). Subsequently in this report, the term "I/O Controller" refers to the device of that name rather than the general set of IOCs.

"IOCs," page 15). The basic CP architecture provides sixteen 32-bit general purpose registers, four 64-bit floating point registers, sixteen 32-bit control registers, four 64-bit Segment Control Registers (SCRs), and a 64-bit Program Control Word (PCW). The CP is equipped with local memory to aid in address translation. This local memory is implemented as either an address Translation RAM (T-RAM), or a Translation Buffer (T-BUF). The CP also supports a Reference and Change Table (RCT), an Arithmetic and Logic Unit (ALU), and a clock. All of these structures will be described in the following paragraphs.

The Register Set

General Purpose Registers are designed to be used as base and index registers during arithmetic and indexing operations, and as accumulators for fixed point arithmetic and logical operations. These registers are identified as **GR0** through **GR15** (**GR15** being the stack pointer) and are specified by a 4-bit field in an instruction. These registers are modifiable by unprivileged users; the unprivileged user has direct read and write access to them.

Floating Point Registers are designed to be used by the floating point and decimal floating point instructions. These registers are identified as **FP0**, **FP2**, **FP4**, and **FP6**, and are only addressable by floating point and decimal floating point instructions. These registers are also modifiable by unprivileged users.

Control Registers provide storage for manipulation of the program control information not contained in the PCW. These registers are identified as **CR0** through **CR15** (see Table 2.1).

Unprivileged users may modify only **CR1** through the use of the following instructions.

- **JUMP TO SUBROUTINE ON CONDITION INDIRECT (JSCI);**
- **RETURN ON CONDITION (RTC);**
- **RETURN AND POP ON CONDITION (RPC);**
- **SUPERVISOR CALL (SVC).**

JSCI provides the mechanism for branching to the entry point of a subroutine. With this instruction the subroutine executes with the caller's context, but could potentially be executing in a different hardware domain (i.e., ring). **RTC** and **RPC** return to the location immediately after the branch made by the **JSCI** instruction.

SVC provides the mechanism for branching to the entry point of a supervisor call. With this instruction the caller's context is saved, the subroutine executes with hardware privilege, and the subroutine could potentially be executing in a different ring. **SVCX** (which is privileged) returns to the location immediately after the supervisor call, after it restores the caller's context (including whatever privilege the caller may have had).

When information is loaded into the control registers, no check is made to determine the validity of the information. The check is made whenever the information is referenced. Hence, an invalid address could be loaded into a control register, but an error would occur when an attempt is made to use that address.

Control Register	Allocation
CR0	**Reserved**
CR1	**Save area back chain**
CR2	**System stack limit word**
CR3	**Debug table descriptor**
CR4	**Prior instruction address**
CR5	**Current instruction address**
CR6-7	**Linkage table descriptor**
CR8	**Pointer to active stack header table**
CR9	**Pointer to stack header table**
CR10	**Debug scope**
CR11	**Reserved**
CR12-13	**Time of day clock**
CR14-15	**Clock comparator**

Table 2.1: Control Registers

SCRs are used to point to tables used in address translation (see "Address Translation Structures," page 11).

Cache Memory Used in Address Translation

Each CP possesses a local page table (LPT) to speed up address translation. The cache memory used in address translation is implemented in the CP's local memory. A LPT contains a subset of the currently executing task's page tables (PTs). SVS/OS implements the LPT as T-RAM (*CP4*, *CP7*, and *CP9*) or T-BUF (*CP8*, *CP10*, and *CP12*). The entries in an LPT can be marked as invalid whenever the **RESET REFERENCE AND CHANGE BITS (RRCB)** instruction is executed (e.g., during a context switch, the operating system would use the **RRCB** to "clean-up" the LPT).

Translation RAM (T-RAM)

The LPT of *CP4*, *CP7*, and *CP9 systems* is called T-RAM. T-RAM entries include memory protection fields that regulate read and write access to the corresponding page frame. The format of T-RAM entries varies for each CP type. The number of T-RAM entries, and the format of these entries varies for each CP type. The number of entries, by default, determines the maximum amount of virtual address space that can be mapped into physical address space, for the given CP. T-RAM entries contain a page frame number as well as a fault-bit and read/write memory protection fields. The *fault-bit* is used to mark an entry as invalid. The memory protection fields are used to regulate read and write access to the page the entry is associated with.

Translation Buffer (T-BUF)

The LPT of *CP8*, *CP10*, and *CP12* systems is called T-BUF. T-BUF is also CP-local RAM and is used to store address translations. T-BUF provides the same functions as T-RAM, but is a different implementation used in the newer CP types. It allows multiple virtual pages to be mapped into one T-BUF entry (i.e., eight mappings). Clearing of the T-BUF is based on a monitor bit and fault-bit. The **RRCB** instruction sets the fault-bit to signify that the given T-BUF entry is invalid. The monitor bit is used for selective clearing of the T-BUF. As with T-RAM, T-BUF also supports read/write protection fields used to determine how a page may be accessed.

Reference and Change Table (RCT)

The RCT is located in CP-local RAM and provides an area to keep track of page changes and page references. There is a 2-bit RCT entry associated with each physical page frame of main memory. A *page frame* is 2048 contiguous bytes (2 KB) of physical memory beginning on a 2 KB boundary. The two bits in the RCT are used to record references or modifications to the given page in memory. RCT entries are cleared by the **RRCB** instruction. When **RRCB** is executed, the RCT entry for a page frame is tested and then cleared.

Arithmetic and Logic Unit (ALU)

The ALU supports the CP by providing mechanisms to perform fixed-point, floating-point, decimal, and decimal-floating-point arithmetic, as well as basic logical operations.

Clock

The time-of-day clock is a 64-bit binary counter located on the CP. System time is kept by incrementing this binary counter.[2] The clock value is stored in **CR12** and **CR13** which are initialized to zero on system power-up. The clock will continue to run as long as it is supplied with power.

Associated with the clock is the clock comparator. It provides a means of causing an interruption when the clock has passed a value specified by a given program (i.e., whenever the clock and clock comparator are equal). The clock comparator is also a 64-bit entity stored in **CR14** and **CR15**.

CP Instructions

The CP provides instructions to perform arithmetic and logical operations. An instruction is from one to four half-words in length and has one of nine basic addressing formats.

The instruction set supports over 220 instructions; a detailed description of these instructions can be found in the *VS Principles of Operation* [27]. There are five general categories of instructions: arithmetic, logical, stack related, miscellaneous, and privileged. Miscellaneous instructions include those used for debugging and other special purposes.

[2]The resolution of the clock, and the rate at which the counter is stepped, is dependent on both the CP type and the line voltage frequency.

Program Control Word (PCW)

The PCW is used in the control of instruction execution. It is 64 bits in length and includes the basic status and sequencing information for the currently executing task. To execute an instruction the CP takes the address of the instruction to be executed, executes the instruction, and increments this address (by the length of the just executed instruction) in the PCW. This process will continue until a branching instruction is reached or there is an interrupt. The PCW is made up of 1-byte of interrupt code, a 3-byte instruction address, a 2-byte system mask, a 1-byte program mask and status information, and a 1-byte field of which three bits are used to indicate the current process level (see Table 2.2).

Bits	Description
0-7	Interruption code
8-31	Current instruction address
32	Wait state
33	Control mode
34	Memory protection and privileged instruction trap
35	Virtual machine mode
36	Reserved
37	I/O interrupt mask
38	Clock interrupt mask
39	Machine check interrupt mask
40	Debug control mask
41-47	Reserved
48-49	Condition code
50	Fixed point overflow mask
51	Decimal overflow mask
52	Exponent underflow mask
53	Significance mask
54-60	Reserved
61-63	Process level

Table 2.2: PCW Description

The *process level* describes the currently executing task's execution level. The level of execution ranges from zero to seven, and represents the privilege level of the given process. The privilege associated with these levels allows the most privilege for process level seven, with privilege decreasing with each level down to zero. A program's privilege level, or bit-34 of the PCW[3], determines whether it may access given areas of memory (see "Memory and Addressing," page 10) or execute privileged instructions.

When bit-34 of the PCW is set, and when the process level is less than seven, the CP cannot execute privileged instructions.

Privileged Instructions

Privileged instructions are those instructions that manipulate the PCW, perform various control functions

[3]When bit-34 of the PCW is cleared there are no traps generated for memory access check violations or the use of privileged instructions. Therefore, the task is executing with privilege equivalent to privilege level seven.

(e.g., **RRCB**), are relevant to address translation, or control I/O. These instructions are considered to be privileged because of the type of actions they perform. Only programs of process level seven, or those with bit-34 of the PCW cleared, can execute privileged instructions.

For a complete description of the privileged instructions see the *VS Principles of Operation* [27].

Use of the Stack

The CP supports separate stacks in a task's virtual memory space for each process level at which the given task can execute. A separate stack for each potential process level is used to limit the possibility of a task, running at one process level, from overwriting the stacks of that same task running at higher process levels. Stack switching is performed when a task makes a call that causes its process level to increase (e.g., **SVC**) or decrease (e.g., **SVCX**). *Stack switching* is the process of activating a new stack. At the time of the stack switch, the system stack vector (i.e., **GR15** and **CR2**) is updated to reflect the status of the stack belonging to the called routine.

Stack switching is implemented through the Stack Header Block (SHB). There is a SHB for each process level and stack associated with that process level. An SHB entry contains the address of the top of the stack, the stack limit, and the most recently built save area. The *save area* is comprised of the context of a task which has made a **JSCI** and **SVC** call. The SHB is accessed by using the value contained in **CR8** to index into the Stack Header Block Table (SHBT). **CR9** points to the SHBT.

When a stack switch involves a call to a higher process level routine (i.e., a **JSCI** or **SVC** call), the following occurs:

- The system stack vector and the **JSCI** back chain address (**CR1**) are saved in the SHB pointed to by **CR8**;
- The address of the called routine's SHB is stored in **CR8**; this address is acquired from the SHBT;
- The system stack vector is updated with the information in the new SHB.

Afterwards, the task is running at a new process level. When control is returned to the calling task (e.g., **RTC, RPC, SVCX**), the following sequence of events take place:

- The current top of stack and **CR2** are saved in the current routines SHB. The SHB is pointed to **CR8**;
- The address of the caller's SHB is restored;
- The system stack vector is restored from the information in the caller's SHB.

Description of Specific CPs

The six CP types (*CP4*, *CP7*, *CP8*, *CP9*, *CP10*, and *CP12*) supported by SVS/OS vary in the low-level implementation of Wang's MLA. The operating system understands and uses only the standard MLA definition. This section provides the reader with a description of the specific CPs. A complete listing of models and CP types is given in "Evaluated Hardware Components," Appendix A, page 66.

The *CP4* uses a T-RAM for LPTs, and interfaces with devices through Bus Adaptors (BAs) in combination with I/O Processors (IOPs).

The *CP7* employs T-RAM and uses high speed caches and virtual addresses at the microcode level to improve performance. *CP7*s interface with devices through Bus Processors (BPs) in combination with Device Adapters (DAs).

The *CP8*s, *CP10*s, and *CP12*s provide the greatest I/O capacities as well as the largest main memory and I/O configuration potential. They interface with devices through microcode-loadable I/O Controllers, use T-BUF for LPTs, and have support for a *Support Control Unit* (SCU) [59]).

*CP10*s and *CP12*s are essentially identical to *CP8*s with increased performance. This performance increase is due to the *CP10* being based on ECL technology and the *CP12* being based on CMOS technology. The resulting hardware differences are handled by the microcode, as is the case with all other CP types.

The *CP9* provides high I/O capacities and memory options spanning from 2 MB to 16 MB. They interface with devices through programmable I/O Coprocessors[4] and use T-RAM for address translation support. Each I/O Coprocessor interfaces with the system bus through its own Bus Interface Controller (BIC). BICs send messages between processors and, during I/O, perform direct memory access between main memory and I/O Coprocessors.

Table 2.3 captures the major characteristics of the six CP types.

CP Type	Cache Support	Maximum Physical Memory	Maximum Virtual Memory	Primary I/O Interface
CP4	T-RAM	16 MB	16 MB	BA & IOP
CP7	T-RAM	8 MB	16 MB	BP & DA
CP8	T-BUF	32 MB	16 MB	I/O Controller
CP9	T-RAM	16 MB	16 MB	I/O Coprocessor
CP10	T-BUF	32 MB	16 MB	I/O Controller
CP12	T-BUF	32 MB	16 MB	I/O Controller

Table 2.3: CP Type Summary

More information on specific CP types can be found in the *VS Principles of Operation* [27], and SVS/OS's processor handbooks [4].

Memory and Addressing

Requests for I/O to main memory can come from the CP or one of the various IOCs. Requests are prioritized with the CP having the lowest priority. Physical memory is divided into 2 KB page frames and is byte-addressable through a 26-bit physical memory address[5]. A physical memory address consists of a 15-bit page

[4] This is a different name, but has essentially the same functionality as the I/O Controller.

[5] Even though a 26-bit address space could theoretically support 64 MB, hardware limitations on the various evaluated CP types allow substantially less than that amount of physical memory. See Table 2.3 for the maximum physical memory for each of the CP types.

frame address, and an 11-bit offset into the given page frame. The range of addressable physical memory depends on the amount of memory for the given CP configuration and ranges from 512 KB to 32 MB.

There is a physical page frame table with an entry for every physical page. Each entry contains information such as the amount of time since last used, whether the page has been modified, file location and usage block (FLUB[6]) pointer, etc.

Virtual Memory

Virtual memory is divided into 2 KB pages and regions and is addressable through a 24-bit address. The *virtual address* consists of a 13-bit virtual page index and an 11-bit byte offset into the given page. The architecture allows for a task's memory to be segmented into 512 regions. A *region* is a group of a varying number of pages beginning on a 2 KB boundary.

At a higher level of abstraction there are segments. *Segments* divide the virtual address space up into unequal quadrants each of which consists of a set of regions. The four segments are identified by SCRs 0, 2, 4, and 6. In the evaluated system, segment 0 represents the first megabyte of virtual address space; segment 2 represents the next eight megabytes of virtual address space; segment 4 represents the last seven megabytes of virtual address space; and segment 6 is invalid (i.e., includes no regions) most of the time. Segment 6 is only made valid, by the paging task (**PAGER**, see "PAGER," page 14), for brief, uninterruptable intervals so that newly allocated physical pages can be accessed by **PAGER**, and then it is rendered invalid again. The only segment that users can directly control the contents of is segment 2; the others are always directly controlled by the OS.

Address Translation Structures

During instruction execution, virtual addresses must be converted to physical addresses. Virtual to physical address translation is performed by the CP. To perform address translation, the CP accesses/manipulates the following structures: SCRs, LPTs, Region Node Tables (RNTs), and PTs.

Associated with each task are PTs. A *PT* is a section of main memory that defines the mapping of virtual address space to physical page frames. There is a separate PT associated with each region of a given task's address space. PTs are logically divided up into 512 16-bit PT entries, representing a maximum of 1 MB. An entry comprises a fault-bit (zero means the page is resident) and a page frame number (if the fault-bit is not zero, this field is ignored). In addition to PT entries, there are two lists associated with each page table. One list contains a bit (i.e., flag) for each page indicating whether it has been referenced and the other contains a flag for each entry indicating whether a paging operation is in progress for that page. The CP keeps an LPT, in its local memory, which contains a subset of the currently executing task's PTs (i.e., entries for the most recently referenced pages) to speed up the address translation process. These on board LPTs are known as either T-RAM (see "Translation RAM (T-RAM)," page 6) or T-BUF (see "Translation Buffer (T-BUF)," page 7), depending on the CP type.

SCRs and RNTs are used by the CP to find the page table for the given reference. An *SCR* comprises three fields: a 4-byte field describing the address range of the corresponding RNT, a 1-byte field describing the size

[6]There is a FLUB assigned to each file in use. It keeps track of how many tasks are using the file as well as what mode of access they have. It also contains information necessary for locating the file on disk.

of the corresponding RNT, and a 3-byte field giving the physical address of the RNT[7]. The RNT address range is described in terms of two 2-byte fields representing the lowest and highest virtual pages included in the RNT.

An *RNT* consists of 1 to 128 16-byte entries known as *region nodes*. Each region node describes a region and points to the LPT for that region. The CP references only the first eight bytes of the region node; the other eight bytes are available for use by the operating system. The portion of a region node used by the CP is logically divided into three fields: the first 4-byte field describes the address range of the region, the next 1-byte field specifies control and access information about the pages in the region, and the next 3-byte field gives the physical address of the PT for the given region. The region address range is described in terms of two 2-byte fields representing the lowest and highest virtual pages included in the region. Two sub-fields of the control and access field are of particular interest; these fields define the minimum read and write access levels for all the pages in the given region (see "Memory Protection," page 14).

The eight bytes of each region node that are not used directly by the CP contain a pointer to a FLUB and an offset indicating where in the file that the region begins. This information is used by **PAGER** to determine where to read pages from and where to write them to. Note that if a file is larger than 1 MB (i.e., the PT limit), it is divided into multiple regions. Each region node will have its own PT, but will point to the same FLUB (but will have different offsets).

Address Translation Procedure

The address translation process begins when the CP references its LPT. If the entry corresponding to the virtual page is found in the LPT, is valid, and the requesting task has the proper access permissions (the LPT entry includes read and write access information), the respective physical address is generated by concatenating the 15-bit physical page frame number in the LPT and the 11-bit byte offset in the virtual address. If there is not a valid entry corresponding to the requested address in the LPT, the CP must perform a complete address translation. If the corresponding entry is valid but the access check fails, a program check is generated and access is denied.

Using numbers to refer to corresponding points in Figure 2.1, the process is as follows.

(1) The virtual page index is compared against the virtual page range defined by the "LOHI" field in the SCRs to find the first segment that contains the virtual page. The SCRs from SCR 0 to SCR 6 will be checked until a match is found or all SCRs have been examined. If the virtual page is not found within one of the SCR defined segments, a program check is generated and access is denied.

(2) The SCR, found in procedure (1) above, is then used to determine the physical address and size (i.e., "RNTSZ") of the corresponding RNT.

(3) The virtual page index is then compared against the virtual page range defined by the "LOHI" field of each region node in the RNT to find the first region that contains the virtual page. The region nodes are checked sequentially until a match is found or all region nodes have been examined. If the virtual page is not found within one of the region node defined regions, a program check is generated and access is denied.

(4) The RNT entry, found in procedure (3) above, is then used first to make an access check using the "MOD" field (see "Memory Protection," page 14). If the access check passes, the region node is then used to determine the physical address of the corresponding PT.

[7] An alternate format for the SCRs is used by **PAGER** to allow it special access necessary for performing paging operations.

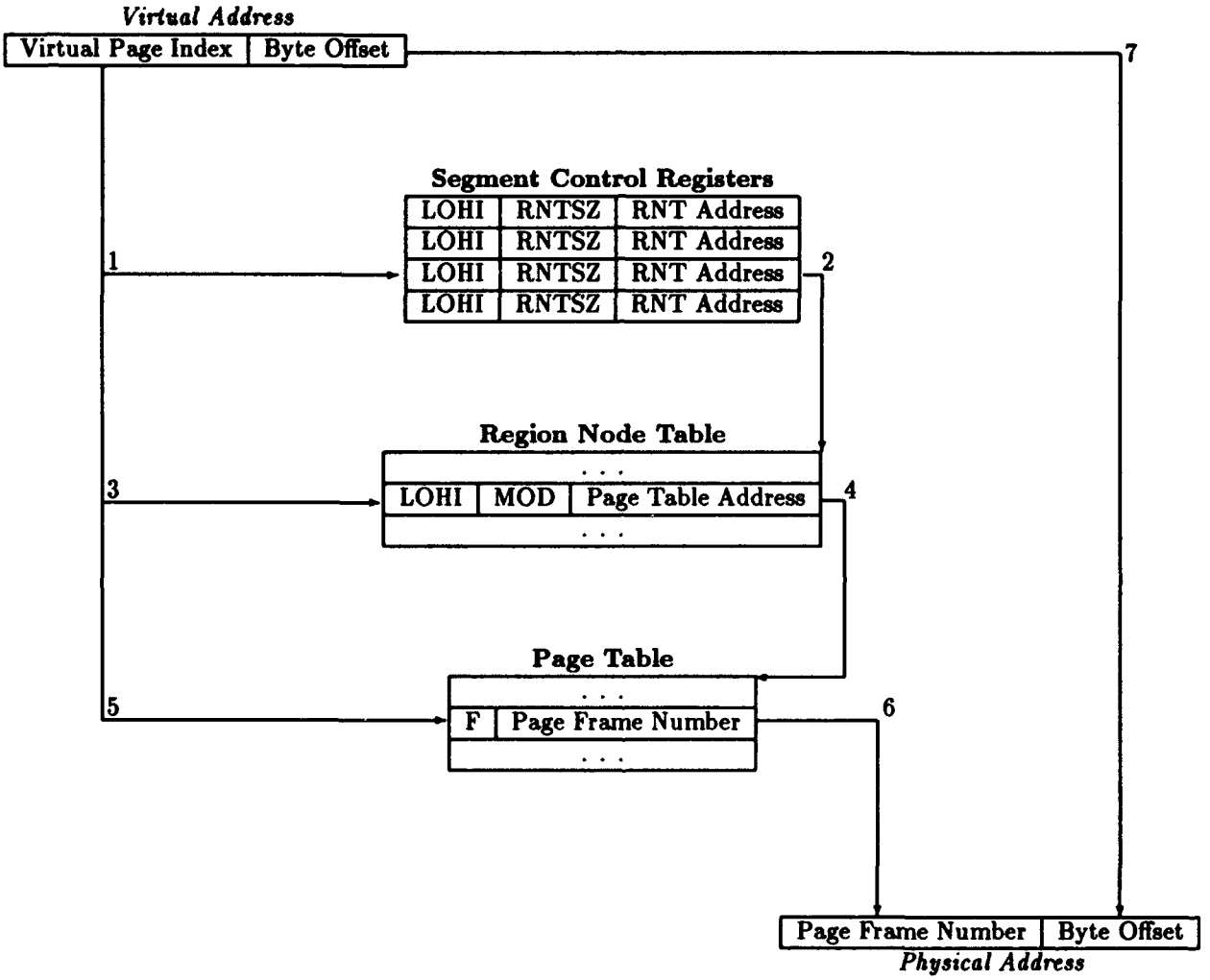


Figure 2.1: Address Translation

(5) The CP then accesses the given PT using an index computed from the virtual page index and the lowest virtual page in the PT (i.e., "LO" from the region node) to find the relevant PT entry. Note that while SCRs and RNTs could have gaps, rendering some of the virtual pages invalid, PTs must consist of contiguous pages.

(6) From the PT entry, the CP can determine whether the given page is valid using the "F" field. If it is not resident, an interrupt occurs and **PAGER** takes over.

(7) If the page is resident, the byte offset from the virtual address is concatenated with the page frame number, and the physical address is returned.

The result (i.e., physical page frame number and access information for the given virtual page) from this address translation is then cached in the LPT for use in future address translations.

PAGER

If a virtual page is found to be nonresident, **PAGER** has to map it to a physical page frame and, possibly, find the given page in a file and load it into memory. This task is privileged and runs with the highest priority.

PAGER keeps track of the fact that it is processing a page fault on a given page so that multiple page faults on the same page are only handled once. It also determines whether the faulted page has been used, and is therefore mapped to a file. If it is not mapped to a file, **PAGER** simply finds a physical page frame and maps the virtual page to it. If it is mapped to a file, **PAGER** must then go into the FLUB, which is pointed to by the region node, and determine where the file is as well as the file offset. It can then move the appropriate page into memory from the file.

When **PAGER** cannot find any free physical page frames, it must make a determination (based on age) regarding which page to swap out. Once **PAGER** finds a page to swap out, it determines whether it has been modified since it has been brought into memory. If has been modified then it is written out to disk using the FLUB (associated with that physical page frame). Next, also using the FLUB, **PAGER** locates the PT entry that is currently mapped to that physical page frame in invalidates it. Now **PAGER** is able to map the virtual address currently being accessed to this now free physical page frame.

Shared Memory

Whenever a file (e.g., a program) is mapped into multiple tasks' address spaces, there is a single PT created for that file. There is also only a single FLUB for that file. However, each task still has its own unique RNT.

Each tasks' RNT will point both to the same FLUB and PT. Within the FLUB, there is information indicating such things as how many tasks are currently using the file and what modes of access are allowed (i.e., certain modes of access require exclusive use of the file). The PT will continue to exist until no task is using that file.

Memory Protection

SVS/OS supports region based read and write memory protection. Associated with every region of virtual address space is an access level. An *access level* is represented by a number from zero to seven, where zero

is the least privileged and seven is the most privileged. Access to a region is determined by a task's current process level. A task running at a given process level can access any region of memory assigned an access level less than or equal to the task's process level.

Access levels are defined separately for read and write access. Included in the entries of the region node table are sub-fields that define the minimum access level required to read and write each individual memory region it describes. These fields are not modifiable by unprivileged users. During address translation, the values associated with these sub-fields are compared with the executing task's current process level (i.e., the process level specified in the PCW). If the task's process level is greater than or equal to the region's defined access level, access to that region is granted. Otherwise, a program check is generated and access is denied. Access level memory protection can be overridden if the given task's PCW bit-34 is not set, indicating it is privileged.

IOCs

IOCs serve as an interface between the CP and various I/O devices. An IOC provides a generic base to describe the SVS/OS I/O architecture; other specific devices will be covered at the end of this section (e.g., BAs, BPs, IOPs, and I/O Controllers).

IOCs execute concurrently with the CP; they are capable of accessing main memory, communicating with devices, and provide the overall support for controlling I/O devices. The structures that allow the CP to interface with the IOCs are the I/O Status Table (IOST), and the I/O Command Table (IOCT).

The IOST is used by the IOC to store the I/O completion status of a given I/O operation. The IOST is made up of 16-byte entries; two important fields in these entries are the I/O Status Word (IOSW), and the Command Table Address (CTA). Information is stored in a given IOSW by an IOC prior to the IOC notifying the CP of completion (via an interrupt). CTA entries point to the IOCT for the IOC associated with the given I/O operation.

There is an IOCT associated with each IOC; it is made up of 16-byte fields (one for each device on the given system). Each entry comprises two fields: an I/O Command Word (IOCW), and a Unit Control Block Address (UCBA). The IOCW specifies the command that is to be executed and is made up of a command code, a data address, and the data length.

I/O Instructions

The basic assembler instructions for I/O are: **Start I/O (SIO)**, **Control I/O (CIO)**, and **Halt I/O (HIO)**. **SIO** is used to initiate a transfer between memory and a specified device. **CIO** is used to initiate control operations on a given device. **HIO** is used to halt an **SIO** or **CIO** request. These three instructions are privileged. Before the CP executes an I/O instruction, it stores command information in the IOCW and notifies the appropriate IOC. The IOC will request service from the given device and wait until the request is acknowledged. On completion, the IOC will update the appropriate IOSW and notify the CP of the completion.

Bus Adaptors

A BA (see Figure 2.2) is an interface device that provides a mechanism for a CP to communicate with one or more IOPs through the system's I/O bus. The BA maintains a pair of registers for I/O between the CP and IOPs. It is used as an intermediary by the CP; communication takes place by the CP sending the BA a message that identifies the desired operation and IOP. The BA will ascertain whether the given IOP is ready to process. If it is ready to process the request the BA passes the message to it. The BA then returns the status to the CP.

Bus Processors and Device Adaptors

A BP (see figure 2.3) performs as an IOC that communicates with a DA. The BP communicates directly with the CP and the DA. The DA provides the functionality to talk to a given device. The BP and DA pair handle all communication between the CP and the devices.

Input/Output Processors

An IOP (see Figure 2.2) is an IOC that provides an interface to a given device and the CP. An IOP is a Programmable Read-only Memory (PROM)-based device. All of its control logic is permanently loaded, in PROM chips, on the interface board.

Input/Output Controllers

An I/O Controller (see Figure 2.4) is an IOC that provides a connection from a CP to a given device. It is similar in function to an IOP, but does not support PROM-based control logic. Instead, an I/O Controller is programmable and is loaded with its controlling microcode (see "System Initialization," page 39) at Initial Program Load (IPL). All I/O Controllers are implemented on a private, high-speed I/O bus which interfaces to the system bus through a System Bus Interface (SBI).

Input/Output Coprocessors

An I/O Coprocessor (see Figure 2.5), which has essentially the same function as an I/O Controller, requires a special interface device (i.e., BIC) since it interfaces directly to the system bus (i.e., there is no local I/O bus). Each BIC supports its I/O Coprocessor with DMA, Interconnect, I/O space, and Message passing operations.

Use of IOCs

Figure 2.2 shows the interactions of the BA and three IOPs. The BA is directly connected to the system bus. The BA provides a local I/O bus, to which the various IOPs are connected. The devices are connected to the IOPs.

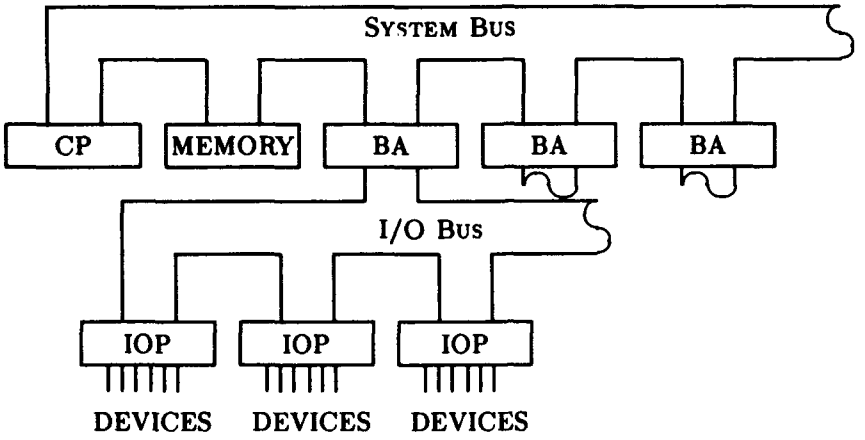


Figure 2.2: Bus Adapter and I/O Processor Usage

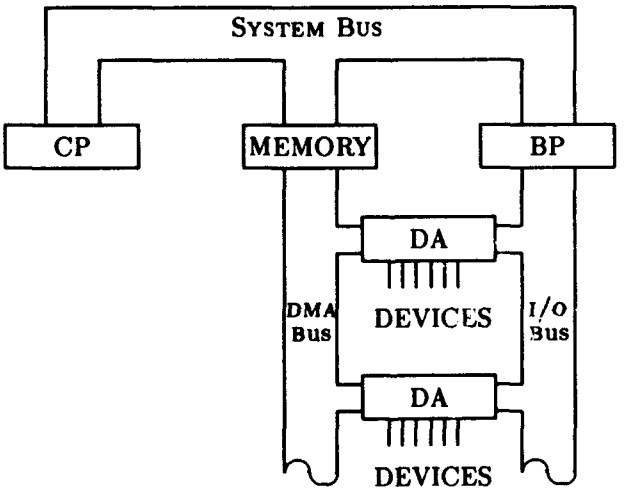


Figure 2.3: Bus Processor and Device Adapter Usage

Figure 2.3 shows the interactions of the BP and two DAs. The BP is directly connected to the system bus. The BP provides a local I/O bus, to which the various DAs are connected. The devices are connected to the DAs; notice that the DAs are also connected to the system I/O bus. This connection illustrates the direct memory access (DMA) channel used by the DAs.

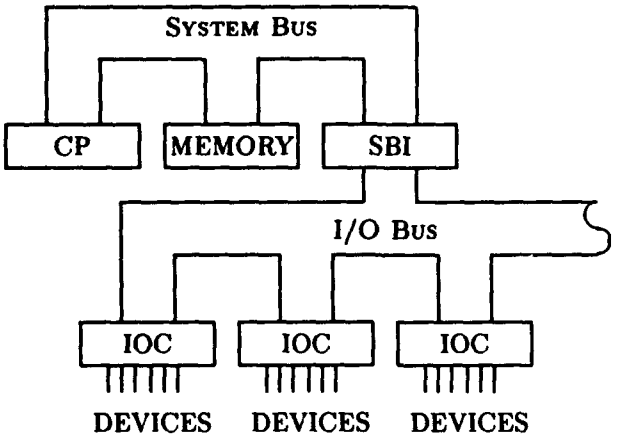


Figure 2.4: I/O Controller Usage

Figure 2.4 shows the connections for an I/O Controller. The I/O Controller is directly connected to the I/O bus and the device.

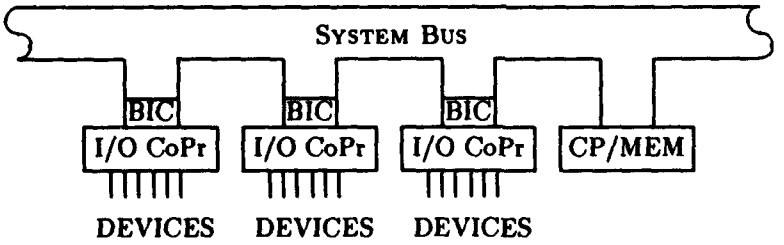


Figure 2.5: I/O Coprocessor Usage

Figure 2.5 shows the connections for an I/O Coprocessor. The I/O Coprocessor interfaces to the system bus through its associated BIC.

Hardware Initialization

The system hardware initialization process differs among the various CPs. The following provides a general overview of the system initialization process for each CP type. More detail on this process can be found in the *VS Principles of Operation* [27].

During a total system initialization the hardware is directly responsible for the following:

- Perform the on-board hardware diagnostic tests;

- Load the CP with its controlling microcode;
- Initialize main memory.

Hardware diagnostics provide a means for the system to ensure that its components are working properly before it begins the initialization process (see "Hardware Diagnostics," page 23). The CP microcode is loaded into the CP's local control store, providing it with its microinstruction set. The clearing of memory is required because the system cannot determine the state of the machine prior to initialization. Since it requires a known memory state, the entire memory is overwritten at this time. Once these three steps have been accomplished, the software takes over.

The *CP4* systems are initialized via a PROM-based IOP. The CP microcode is loaded from the IPL device (i.e., a local disk drive which is designated for this purpose). IPL is achieved by loading the microcode from the local disk and pressing the boot strap button on the system front panel. This leaves the machine in control mode (see "Control Mode," page 32). Load commands are used, from the system console (*Workstation 0*), to initiate the loading of the software, to specify the BA, the port of the IPL device, and IOP.

The *CP7* systems do not support any load commands. Loading of the microcode is accomplished by manually selecting the diagnostic IPL device that provides a mechanism to select the IPL boot device. Microcode loading is performed from *Workstation 0* via a menu driven application program. The BP controls IPL up to the point when Control Mode is reached.

The *CP8*, *CP10*, and *CP12* systems are initialized from a Wang Personal Computer (PC)-based SCU. By turning on the SCU, the system is initialized and left in Control Mode.

The *CP9* systems are initialized from Workstation 0, which is attached to the system bus through a special, non-optional I/O Coprocessor known as a Resource Control Unit (RCU). The RCU orchestrates the IPL process, loading the operational and CPU microcode required for system generation: this includes both boot-up and diagnostic software.

Support Control Unit

The *SCU* consists of the system panel and the workstation that is cabled as device 0 on port 0. For *CP8*, *CP10*, and *CP12* processors, the SCU is the "System Console." The operator can cause the CP to enter Control Mode manually by using the SCU (*CP8*, *CP10*, and *CP12* systems only) or the Control Mode button on other CP systems. The ability to enter control mode can be controlled by a key lock on all CPs, except *CP4*.

The SCU is a 16-bit microprocessor-based unit with a built-in $5\frac{1}{4}$ inch diskette drive and a fixed-disk drive. The SCU serves as Workstation 0, and can also function as a conventional workstation. The SCU has direct access to the *CP8*, *CP10*, or *CP12* processor, independent of the system bus, thereby allowing it to access the system even if there are malfunctions of major system elements.[8]

The SCU can perform the following functions:

- microcode and diagnostic code loading;

[8] The SCU runs a special applications software package known as SYSCON. SYSCON, written by Wang, runs on a stripped-down version of MS-DOS, and is supplied with the SCU.

- initial program load (IPL);
- memory dump operations;
- local and remote[9] maintenance, including hardware and software support;
- standard workstation operations.

2.1.2 Devices

The Wang VS Product Family supports four types of devices: workstations, printers, disks and tapes. For more detail refer to the *VS Principles of Operation* [27]. The evaluated configuration consists only of devices that are directly connected to the host system.

In general, all devices communicate using an IOCW from the CP to the device and an IOSW from the device to the CP. The system supports indirect addressing through a data address which points to a list of addresses of buffers to be used in the data transfer.

IOCs are loaded with microcode, by the operating system, via the LOADCODE SVC. This interface downloads a microcode image file that provides a controller with its micro-instruction set. The loading of this microcode takes place at system initialization (see "System Initialization," page 39) or when requested by an authorized user (e.g., operator). During system initialization all I/O devices are reset; a reset causes the device to terminate all ongoing operations, and wait for its microcode. The system will generate a process, @IOSLOAD, which reads the configuration file that specifies the list of system devices (i.e., @CONFIG@). From this list the location of the microcode image files for the given IOC, and the appropriate control parameters (e.g., bus location, transfer locations) are determined. Once this information has been determined, @IOSLOAD loads all IOCs with the appropriate microcode image and control parameters.

Workstations

SVS/OS supports workstations with either 24 or 25 lines of 80 characters each. Refer to "Workstations," Appendix A.3.3, page 74, for a complete list of supported workstations. Line 25 is supported on many of the workstations. This line is a message/status line that can be used by an application program to communicate application dependent information to the user at the workstation. All supported workstations conform to a common architecture that is described in the following paragraphs.

The workstation keyboard, called the "Wang Universal Keyboard" has a multitude of keys for data entry, cursor positioning, system communication, and other special purpose keys.

These workstations support the ASCII standard character set as well as the FIPS-15 95-character subset standard [61]. The workstation screen can be formatted into distinct fields, each having a distinct set of attributes. The attributes control the display form and intensity of a field, and the ability of the user to modify it.

The buffer area specified on an I/O transfer is called a data area and is made up of a 4-byte order area and a mapping area. The *order area* contains control information (i.e., screen location). The *mapping area* contains the data and field attribute characters.

[9] Since remote maintenance requires connecting the SCU to a phone line, it is not considered part of the evaluated configuration.

The workstations support the following commands:

- **READ** - copy specified part of screen to the mapping area and store the cursor position in the order area[10];
- **READ ALTERED** - read fields that have the selected-fields tags set into the mapping area;
- **READ DIAGNOSTIC** - same as the READ command except that it does not affect special fields on the screen (i.e., pseudo-blanks are not converted, field attributes are left unchanged, and the cursor position is left unchanged);
- **READ TABS** - read the column numbers of all the set tabs (up to ten);
- **WRITE** - transfer the data from the mapping area to the screen;
- **WRITE SELECTED** - transfer from the mapping area those fields that have the selected-field tags set to the screen;
- **WRITE TABS** - causes all tabs to be cleared and then sets up to ten tabs as specified from the mapping area.

When a workstation is powered on, an attention interrupt is generated and the system responds by loading the microcode for that workstation. Refer to "System Initialization," page 39, for more information on the details of this process. Additionally, microcode can be subsequently loaded from SVS/OS to the workstation at any time. This allows users to customize the "personality" of the workstation, and SVS/OS to restore the system default microcode when workstation ownership changes.[11]

Printers

SVS/OS supports a full range of printers. Refer to "Printers," Appendix A.3.4, page 74, for a complete list of supported printers.

Data transfer is based on variable length blocks of up to 2 KBs. The following block types are supported:

- **PRINT DATA BLOCK** - contains records to be printed;
- **PRINT CONTROL DATA BLOCK** - contains records used for controlling the printing of print data blocks (e.g., vertical and horizontal pitch, form length and printer speed);
- **FONT DATA BLOCK** - contains font files. The default font is loaded into the printer when the system is initialized or when the device is powered on. Other fonts may be loaded in response to a request from the application using the printer;
- **IPL CODE OVERLAY BLOCK** - contain overlays of printer microcode. One overlay is loaded into the printer when the system is initialized or when the device is powered on. Other overlays may be loaded later in response to a request for a special function from the application using the printer.

[10] The **READ** includes converting pseudo-blanks to real blanks and changing blinking fields, of the screen, to high-intensity.

[11] Since the TCB depends upon some of the functions of such workstations to enfore its security policies (e.g., object reuse), it is necessary that these workstations be protected from unauthorized tampering or modification.

All systems support font and IPL code overlays. On power up, after the microcode is loaded, the printer will request the font and IPL code overlays in a IOSW. For more information on this process see "System Initialization," page 39. The complete protocols for the printers can be found in the *VS Principles of Operation* [27].

The combination of the command codes and the sub-command codes define eight specific functions that can be performed. These are:

- **PRINT DATA BLOCK** - transfer a print data block to the printer to be printed;
- **CONTROL DATA BLOCK** - transfer a control data block to the printer;
- **READ INFORMATION** - this function is sent to the printer in response to a font load request. It specifies a memory location for the printer to store the font identifier for the font to be loaded;
- **POWER UP** - this function tells the printer whether the system supports IPL overlay code and font loading, as well as the CP type. This function is sent after the printer is powered up and loaded with microcode. The printer then requests the loading of the IPL code overlay and the default font;
- **ERROR** - this functions tells the printer that the system could not honor the request for the load of a font or IPL code overlay;
- **IPL CODE OVERLAY** - this function is used to transfer microcode to the printer;
- **FONT DATA BLOCK** - this function is used to transfer font files to the printer;
- **END OF JOB** - this function is sent to the printer after the printer acknowledges the last print data block for the job.

Disks

Wang supports a variety of disk devices including both fixed and removable devices. Refer to "Disk Drives," Appendix A.3.1, page 71, for a complete list of supported disk drives. The standard logical sector size supported across all disk devices is 2 KB. The memory area used to transfer data from/to a disk must be one or more pages. The starting sector address plus the data count cannot imply a cylinder change. The disk devices support auto-retry and indirect addressing. With indirect addressing the address in the IOCW points to a list of addresses to be used on the I/O operation.

The following commands are supported by the disk devices:

- **READ** - read information from the disk;
- **WRITE** - write information to the disk;
- **WRITE VERIFY** - used by the controller to verify the data that was written on the disk. The data is first written to disk. The controller then reads the original data from memory and reads the data from the disk. The data is compared to ensure that it was written correctly;
- **READ VERIFY** - used by diagnostic programs to test the verification logic. It performs all but the initial write of the write verify operation;

- **SEEK** - position the access mechanism;
- **FORMAT** - format the addressed sectors with the sector preamble;
- **RELEASE** - supported for dual port devices only. It causes the controller to give up the reservation of a dual port disk;
- **RESERVE** - supported for dual port devices only. This command causes the controller to attempt to reserve a dual port disk. I/O to an unreserved, dual ported disk will cause an implicit reserve.

Tapes

Wang supports a variety of tape devices that include normal $\frac{1}{2}$ inch reel-to-reel tapes and $\frac{1}{4}$ inch cartridge tape. Refer to "Tape Drives," Appendix A.3.2, page 73, of for a complete list of supported tape drives. The reel-to-reel tapes support both 7- and 9-track tapes with parity and longitudinal redundancy checks while 9-track tapes also support a cyclic redundancy check. The cartridge tape device records data serially on one of four tracks with every ninth bit being a parity bit. Each tape block also uses a cyclic redundancy check.

All of these devices support auto-retry (backspace and rewrite/reread) and indirect addressing. In indirect addressing the address in the IOCW points to a list of addresses to be used in the I/O operation.

All tape devices support the read and write operations and control functions that are standard on most tape devices in the industry (**SENSE**, **ERASE TAPE**, **WRITE TAPE MARK**, **FORWARD SPACE FILE**, **FORWARD SPACE BLOCK**, **BACKWARD SPACE FILE**, **BACKWARD SPACE BLOCK**, **REWIND**, and **REWIND/UNLOAD**).

Additionally there are commands that are supported only by specific devices. The **FIND TAPE LENGTH**, **SET WRITE CURRENT HIGH**, and **SET WRITE CURRENT LOW** commands are supported on the cartridge tapes since the 600-ft. tapes are thinner and require a higher recording current than the 300- or 450-ft. tapes.

The **Set Density** command is supported only on dual- or tri-density 9-track devices. The **Set Parity** command is supported only on 7-track tapes.

2.1.3 Hardware Diagnostics

The SVS/OS system provides diagnostic support for checking the correct operations of its CPs, IOCs, memory, controllers, and peripheral devices. These diagnostic support tests are used in the development cycle of a given component, as well as for online system diagnostics.

The primary focus of the hardware diagnostic tests is centered on the CP. During the development cycle of the CP, architectural tests are run to ensure that the given CP meets the SVS/OS standard (i.e., the hardware conforms to the MLA). The VS Basic Assembly Language (*BAL*) test suite is run on every CP type to ensure that it correctly implements the SVS/OS instruction set. These tests are aimed at checking the proper function of all the instructions supported by the architecture.

The BAL tests are broken into three functional areas:

- **VS Extended (VSE) BAL** instruction tests;

- **Span BAL** instruction tests;
- **Floating Point BAL** instruction tests.

The entire test suite (i.e., all three components listed above) is collectively known as *SUPER-BAL*.

VSE BAL instruction tests ensure that the basic instruction set, described in the *Principles of Operation* [27], functions as specified. The individual tests test the instructions for correct operation, error returns, side effects, and condition code setting. The entire instruction set is tested with each test ensuring that:

- Correct results are produced and stored into the appropriate registers and/or memory locations;
- Correct condition codes are set or ignored as is appropriate;
- Registers and memory locations not involved in the given instruction test are left unaltered;
- Boundary condition tests return the expected results;
- Program exception cases occur when appropriate.

Note that certain instructions are not specifically tested (e.g., **Load**, **Store**, **Branch**, and **Compare Logical**).

The **Span BAL** tests are aimed at testing the CP microcode that is involved in performing operations in which either one or both of the operands span page frames in memory. Not only do these tests ensure that the instruction is performed correctly, but also they ensure that the correct reference and change bits remain cleared for unreferenced pages. Also part of the span tests are tests which check the proper restorability of instructions that are interruptible.

The **Floating Point BAL** test suite tests the floating point instructions. This testing is accomplished by a 4-part procedure applied to all floating point operations:

- A generalized instruction test loop;
- General input pattern generation routines;
- Special case input routines;
- An emulation algorithm for the instruction being tested.

The results of these test are compared to the expected results, and errors are reported to the user.

It should be noted that the *BAL* and *SUPER-BAL* tests are used internally by Wang to verify that new CP types conform to the *VS Principles of Operation* [27] and the MLA interface. These tests are not shipped to customers.

Each of the SVS/OS systems (except those based on *CP4*) is equipped with power-on diagnostics that run as the initial step of IPL. These diagnostic tests check the status (i.e., availability and functionality) of the various components connected to the given system. An error code is returned to the console when a test fails. If an error is determined to be system critical (i.e., the system could not run without the given component) the system halts.

A VS system is also provided with stand-alone diagnostics that can be invoked at System Administrator (SA), operator, or diagnostician discretion (see "User Interfaces," page 30). These tests consist of a subset of the **VSE BAL** test suite. Wang also provides field engineer support for extended offline diagnostics.

2.2 Software Architecture

The software Trusted Computing Base (TCB) consists of the kernel software and the following utilities: **BACKUP**, **BUILDALT/SORT**, **DISKINIT**, **FLOPYDUP**, **GROUPEDT**, **SLFORMAT**, **SLPRINT**, **SYMBOLIC DEBUGGER System Services**, **TAPEINIT**, **VOLCOPY** and **VSSECURE**. The following sections describe the functions performed by this software.

2.2.1 Kernel Software

The kernel software is divided into three parts: the nucleus, the Data Management System (DMS), and system services. The *nucleus*, maintained in the file **@SYS00n@**, where n is the CP type, consists of that software which must be resident or is called by a routine in the nucleus. The DMS routines reside in the file **@SYSDMS@**, and the remaining routines reside in the file **@SYSSERV**.

2.2.2 Task Execution

The basic unit of work in SVS/OS is the task. A *task* is similar to what is called a process in many other systems, and includes the environment within which users and the system perform functions and run programs.

Task status and context information is contained in two structures: the *task control block*, and the *extended task control block*. Tasks communicate with each other through main memory, secondary storage, or by using inter-process communication (IPC) mechanisms (see "Interprocess Communication," page 36).

There are three categories of tasks: system tasks, user tasks, and initiator tasks. *System tasks* can be either resident or non-resident. *Resident* tasks are those tasks whose code and data are permanently fixed (resident) in memory. The resident system tasks are the I/O monitor and the pager (**RITA** and **PAGER**). **RITA** and **PAGER** always execute in a privileged state (i.e., PCW bit 34 = 0).

Non-resident system tasks (also called *dedicated* system tasks) include activities such as task, queue, and printer management. They generally run in user state (i.e., PCW bit 34 = 1, and process level = 0). However, they can call SVCs that normally require the caller to be privileged, or at least be executing at a process level ≥ 2. The operating system can determine that a task is a system task by checking a flag in the task's task control block (see "Task Management," page 27).

There are two types of *user* tasks: foreground user tasks, and background user tasks. A *foreground user task* is created when a user logs on to the system, and all command and program execution occurs within that task; the task terminates as part of the logoff sequence. *Background user tasks* are non-interactive, execute without user intervention, and can be invoked by foreground or other background tasks.

Initiator tasks are background tasks. They are created and started by the system operator and exist until

terminated by the operator. They are essentially empty tasks, to which work is assigned as required. They are used to run background work on behalf of requesting users, executing with each requesting user's identity, in turn.

One task can create another task, which then becomes a dependent of the originating task. The terms *parent* and *child* are used to describe the relationship between tasks. A parent task and its child tasks are collectively referred to as a *family*. A child task may be either a foreground interactive task, or a background non-interactive task. If it is interactive, the workstation associated with the parent task can be released from the parent and passed to the child task. Parent tasks can check for completion of execution of their progeny, and can terminate them if they wish. Each task has its own assigned virtual memory (see "Task Virtual Memory," page 26). The Task Manager is the root node of the task "family tree."

The *link level* (or program) mechanism is used for sharing work among tasks. It is similar to one program calling another. The **LINK** and **UNLINK** system services provide creation and termination of link levels within a task. Each link level operates within the same virtual memory address space as the task of which it is a part, and execution occurs in only a single link level at any one time. Each link level is represented by a data structure (called the program file block) and a save area. Any files or program libraries introduced into the address space at one link level are deleted when the program returns to its calling link level. The invoking link level must have at least "execute" access to the program file it is attempting to link to. The command processor initiates user programs, when the user issues a run request at the workstation, by using **LINK**.

If the user presses the **HELP** key during execution of a program, if the program issues a **CANCEL** system service call, or if the program encounters a program check, the user is given the option of using the Debugger to examine that program, or of continuing or terminating the program. The SA can deny the user the capability to invoke the Debugger, in which case the above conditions would lead to abnormal termination of the user's task.

2.2.3 Task Virtual Memory

A task's virtual memory (see Figure 2.6) is divided into two areas: operating system and user area. The total size of the virtual memory is 16 MB: the user and operating system areas are each 8 MB in size.

The *operating system memory* consists of 14 regions (see "Virtual Memory," page 11). One of these regions contains the nucleus code, several others contain various types of operating system data areas, and the remainder contain stacks, one for each process level (see "Process Levels," page 29). Of these 14 regions, all but five (the four stacks and the user's linkage table) are common to all tasks. All the operating system areas are write-protected from the user task; in addition, the stacks and the interprocess message area are read-protected from a user task.

The *user area* is divided into the program code area, the user Modifiable Data Area (MDA), and the area between these two. The program code area and the user MDA each consist of a single region. The size of the user MDA is fixed for each user by the SA (using the **VSSECURE** utility). The user MDA is used for the stack and buffer (or heap) areas. The stack and heap grow from opposite ends of the user MDA; the TCB maintains CP registers (see "Use of the Stack," page 9) which contain the current addresses of the dynamic boundary of each of these areas, and signals an error if the two meet.

The size of the program code area is determined by the size of the program being executed. This area is used only for reentrant code, and is write-protected, which means that one copy of the code can be shared by all the users who wish to execute it.

The area between the program and user MDA (the "MSMAP" area in Figure 2.6) is used to map program and data files directly into a task's virtual memory, so that the program does not need to request I/O operations to move the data into and out of memory; the paging mechanism is used instead. Each program or data file that is mapped into this area is assigned its own regions: one, which is writable, for data, and the other, which is not writable, for code. Such files may be opened in exclusive or shared mode, thus offering an additional method by which tasks can share code and data. Mapping and unmapping of files is auditable. Note that the region is the smallest unit of memory which can be shared.

Since the paging mechanism does not ensure that the most recent modifications to a memory-mapped data file are written to disk, Wang cautions users that files mapped into the MSMAP area should only be read and not modified; the area can reliably be used to hold modifiable data when such data is used for interprocess signalling rather than for longer-term storage.

The TCB maintains a number of structures for managing the files opened or mapped by a task. Every file in use by a task (or by the operating system) has a FLUB, pointed to by a hash table in system memory. Every time a file is opened, an Open File Block (OFB) is created. The OFB points to the FLUB for that file. All OFBs associated with a task are chained together, and linked to the task control block. Each program creates a User File Block (UFB) for each file that it opens. The UFB and OFB point to each other. Dynamic access control to a file is performed using information stored in these structures.

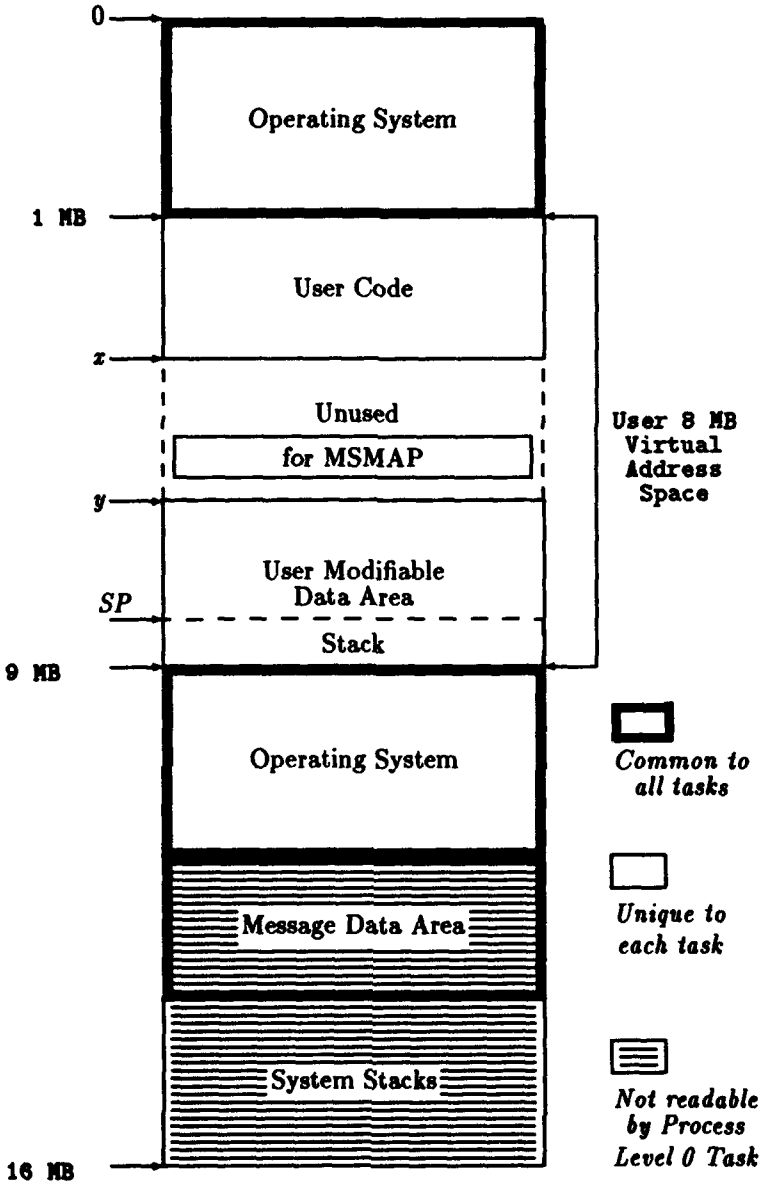
2.2.4 Task Management

SVS/OS creates system and user tasks dynamically. From the time a task is created until it terminates, it exists in one of four *states*: Active, Runnable, Waiting (blocked), or Suspended; the first three states have the customary meaning, while the last state indicates that the task is stopped because of an unexpected error.

Task scheduling and task switching are controlled by the TCB, using priority information associated with each task. Task scheduling is performed by the Task Scheduler and task switching is performed by the Dispatcher. The scheduling algorithm uses conventional multiprogramming techniques to provide variable time slices to different task types. Sixteen different priorities are assigned to tasks, with the six highest assigned to system tasks, and the remaining ten to user tasks.[12] While system tasks retain a fixed priority and a fixed time slice,[13] user task priorities and time slices depend on the type of task and its current usage patterns. Word processing foreground tasks are given the highest initial user priority, while the initial priority for other foreground tasks is lower, and that of background tasks is the lowest initial priority of all. If a task exceeds its time slice, it is assigned to the next lower priority, and given a longer time slice when it next executes. However, the lowest priority given to a task is dependent on the task type; for example, a foreground word processing task will never have a priority lower than a low priority task. The I/O monitor task (**RITA**, see "Meter Maid (**RITA**)," page 35) checks for queue stagnation and promotes low priority tasks until they are given a chance to execute. In addition to time slice interrupts, tasks may be interrupted by I/O or external sources; the Scheduler is used to decide, on each such interrupt, what task should next be executed. The Dispatcher, which is called only from the Scheduler, then selects the highest priority runnable task and initiates its execution.

[12] All but the top two priorities can be assigned to a user task by the SA.

[13] The highest priority is assigned to the kernel itself, which runs with an unlimited time slice.



x: Determined by size of user code

y: Fixed for each user; determined by SA

SP: Stack Pointer

Figure 2.6: Virtual Memory Map

Task Manager (TSKMGR)

TSKMGR is a background dedicated system task which manages interactive workstations. It creates the operator task for the system console, generates and deletes interactive tasks, supplies appropriate screens to workstations, and handles logon activities. After the logon process is complete, **TSKMGR** spawns a child task and passes ownership of the workstation to that task. When the child task terminates, **TSKMGR** takes control of the workstation.

2.2.5 Process Levels

User tasks can invoke TCB services by making direct calls to the TCB (see "System Services," page 32). The system's process level mechanism is used to protect the TCB from user tasks. Each task has five distinct *stacks*, each of which corresponds to a different process level at which the task can operate. The process levels at which a task may run, and the software it primarily executes in those levels, are:

0 — user program execution and debugging;

1 — Command Processor/Operator Interface and OS interface to debugger;

3 — input/output tasks (see "The Data Management System," page 34);

6 — SVC-type system services;

7 — JSCI-type system services, and the nucleus.

Each such stack is a separate region (see "Virtual Memory," page 11), with its own assignment of access level (see "Use of the Stack," page 9). A system stack is thus maintained in a task's virtual address space for each process level at which the task may execute. Hence, a task executing at process level 0 cannot access data placed on the stack by TCB routines operating at, for example, process level 3. Each stack supports a buffer (heap) area, so the TCB routines can use this area for data which is to be kept isolated from direct access by the user task.

2.2.6 Wait Levels

User tasks are assigned to one of two *wait levels*. There is a flag in the task control block which indicates, at all times, which wait level the task is currently in.

Wait level 1 is the *user* wait level. Wait level 1 is used whenever a user is executing any untrusted code in process level 0. Wait level 2 is the *command processor/operator interface* wait level. Wait level 2 is used whenever a user is executing the command processor or debugger, both of which are trusted. Since there are exactly two wait levels available to a task at any given time, a task essentially maintains two program contexts, one being trusted and the other being untrusted. There is a flag in the task control block that indicates which wait level the task is currently in at all times.

The user's task enters wait level 2 when the user presses the **HELP** key, the user program encounters an unexpected program interrupt, on completion of a program initiated from the command processor menu, and immediately after the initial user logon (if no initial run file is automatically executed).

In actual use, a task spends most of its time in wait level 2 (the command processor) where a number of functions are provided for the task by the trusted code. When a task executes something untrusted, the transition to wait level 1 occurs and an entry address is stored for wait level 2. This address will be used on a return to wait level 2. If the program was interrupted to get back to wait level 2, an entry address is saved for wait level 1. This entry address can be used from the command processor to continue executing the wait level 1 program.

Hence, each task is essentially running two sub-tasks (one trusted and one untrusted) which are switched back and forth through well defined mechanisms.

2.2.7 Job Processing

Jobs are background tasks submitted for processing using the **SUBMIT** command or SVC. These jobs, for which a procedure file (see "Command Processor/Operator Interface," page 31) is provided, are placed into a job queue which is managed by the System Task, **SYSTSK**. **SYSTSK**, after selecting a job from the job queue, assigns it to an available initiator task for execution as a background task. At job completion, the initiator task becomes available for assignment to another job. The job submitter can specify when the job is to be executed and the file class default (installation specified). Job submittal is also used to queue print files for printing.

2.2.8 User Interfaces

A user is defined as a person logged on to a workstation and can be a SA, operator, diagnostician, or unprivileged user.

The SA is a user who has been granted System Administrator privilege. This privilege gives the SA access to all files on the system and the ability to completely control the system.

The operator is a user who has been granted access to the Operator Interface menu from any workstation.

The diagnostician is a user who has been granted diagnostic privilege. This privilege allows the user to execute disk I/O diagnostic programs. Such a user can potentially access all data on the system.

Unprivileged users are users who have not been granted any of the specific privileges detailed above.

SVS/OS provides three separate interfaces between a user and the TCB: the Command Processor/Operator Interface, System Services, and Control Mode. The *Command Processor/Operator Interface* is available to all four user roles, although some of its capabilities are different for different users. The *system services interface* is available to tasks operating on behalf of any of the four types of users; some of the system services will be more restrictive for unprivileged users than for the other roles. The *control mode interface* is available only to an operator at Workstation 0.

Command Processor/Operator Interface

SVS/OS is menu-driven, and makes extensive use of a workstation's screen formatting features and Program Function (PF) keys. The Command Processor main menu is displayed after logon to SVS/OS, as well as after the completion of any program, procedure, or command. A user communicates interactively with the

system by selecting commands from this menu, and any subsequent menus which appear in the course of the user's interaction with SVS/OS. Program support facilities exist for the creation of menus to be displayed, and for the acceptance of data provided by the user. These facilities also support program- and file-provided data, so that an application can be developed that will operate equally well whether used interactively, or invoked as part of a non-interactive procedure.

The SA can restrict the user's access to the main Command Processor menu or to certain Command Processor functions.[14] The SA can further restrict a user's access by specifying a particular logon program or procedure to be invoked as part of that user's logon process. If the SA restricts a user's functions, the Command Processor menu displays only the functions that the user is permitted to perform.

The functions available to the user from the main Command Processor Menu are as follows:

- RUN program or procedure;
- SET usage constants (e.g., default values, environment variables);
- SHOW program completion report;
- Manage QUEUES;
- Manage FILES/LIBRARIES;
- Manage DEVICES;
- Enter WORD PROCESSING;
- SUBMIT procedure;
- Send MESSAGE to operator;
- PRINT COMMAND screen;
- LOGOFF.

A user with Operator privilege will have the additional option, "Enter OPERATOR mode," available at the main menu.

The modified Command Processor menu, which appears when the user presses the **HELP** key, is similar to the main Command Processor menu, except that it allows the user to enter the Debugger, continue execution of the interrupted task, or print the program screen.

A separate menu family is available to an operator. The first menu in this family appears when the user presses the key to select the OPERATOR mode from the Command Processor menu.

The user can execute system commands using a *procedure file*, which is a file containing system commands, user parameters, and statements unique to the procedure language (described in the *VS Procedure Language Reference* [15]). The user can execute a procedure file by invoking the procedure language interpreter directly from a workstation menu, or by using the **SUBMIT** command to queue the procedure file for later execution as a background task.

[14] However, a user who is able to create and compile programs can still access some of these functions through the System Services interface (see page 32).

System Services

The program interface to SVS/OS is implemented in a large number of routines called *system services.* They are separated into those that can be executed only by privileged tasks, and those that can be executed by any task. Several of the services that can be executed by unprivileged tasks have options available only to privileged tasks.

The two forms of system services are:

- JSCI-type system services — these services use the **JSCI** instruction to call an individual service routine. These services can be called from assembly-language or high-level-language programs;
- SVC-type system services — these services use the **SVC** instruction to call an individual service routine. These services can be called only from assembly-language programs.

Refer to "Use of the Stack," page 9, for a description of the **SVC** and **JSCI** instructions.

The types of system services available are:

- program services, including task initiation and termination, timing, and interrupt handling;
- I/O services, including the allocation of resources to requesting tasks, the control of peripheral devices, and the loading of microcode to devices;
- memory management services, including the dynamic allocation of buffer storage and shared memory data and code areas;
- communication and synchronization services, including intertask communications and data sharing;
- file services, including file and disk space management;
- security services, including protection of data structures and tasks, management of the audit trail, and allowing users to control the sharing of their data.

Control Mode

Control Mode is a hardware state of the CP in which normal program execution is halted, and other functions are made available to the Operator. These facilities are implemented by two groups of commands:

- *Load Commands* — commands for loading the operating system or a standalone program into main memory;
- *Debug Commands* — commands for displaying or modifying main memory, registers, and the PCW; for controlling single-step program execution; and for controlling virtual address translation.

Control Mode commands are entered at Workstation 0 on *CP4*, *CP7*, and *CP9* systems, and at the SCU (running in Control Mode) on *CP8*, *CP10*, and *CP12* systems.

Control Mode can be entered during program execution or at IPL time (see "Hardware Initialization," page 18). When Control Mode is entered at IPL, both load and debug commands are available. When it is entered during program execution, only debug commands are available.

A system enters Control Mode during program execution after any of the following events:

- the Control Mode bit (bit 33) of the PCW has been set;
- the Control Mode button on the system control panel has been pushed (*CP4*, *CP7*, and *CP9* systems);
- Control Mode has been selected from the console menu (*CP8*, *CP10*, and *CP12* systems);
- a single-step trap, set from Control Mode, has been taken;
- a machine check has occurred and the M-bit of the PCW (bit 39) has been reset, disabling machine-check interrupts.

Each of the above conditions depends upon access to the system hardware, or execution of a privileged task (to modify the PCW).

2.2.9 File System

SVS/OS defines a *record* as a collection of bytes up to 2 KB in length, and a *file* as a logical unit consisting of zero or more records of related information. A file is assigned a name that consists of one to eight characters.

The SVS/OS file system structure is composed of files, libraries, and volumes. A *library* is a collection of files identified by a library name, which can be one to eight characters in length. Library and file naming conventions are the same. A *volume* is an independent storage device (e.g., disk, tape) which can contain one or more libraries. A volume is identified by a volume name that can be one to six characters in length. Volume and file naming conventions are the same with the exception of the length attribute. A *volume set* contains one or more volumes and allows very large files to span two or more volumes.

Three types of files are supported by SVS/OS: output files, print files, and work files. *Output files* are user-created disk files, such as programs, procedures and data files. *Print files* are disk files that contain the output that a program sends to a printer. They contain printer control characters needed by printers for such functions as font control and line feed control. The system stores print files within the user's print library, unless otherwise specified. *Work files* are temporary disk files that the system uses during program editing, compilation, and execution. They are used for internal processing only. SVS/OS creates these files and deletes them after using them. The system stores work files within the user's work library.

SVS/OS supports several types of file organization:

- consecutive;
- indexed;
- relative;
- WP;

- *program*;
- print.

File attributes are described in detail in "Files," page 44.

2.2.10 The Data Management System

DMS is the component of SVS/OS which handles file operations and space allocation on external storage devices, including disks and tapes. DMS performs all I/O operations and controls the physical location of information on each volume.

Data Organization on Disk Volumes

All disk volumes associated with SVS/OS are owned by the SA and operators.

A disk volume is divided into a number of physical units called blocks; each block is 2 KB in length. When a file is created, the DMS requests, through a system service, space for the file by reserving a number of contiguous blocks on the disk volume. If a file outgrows its initially allocated space, DMS automatically allocates additional space on the volume and assigns it to the file. A group of contiguous blocks allocated for use by a particular file is known as an *extent*. A *Volume Table of Contents* (VTOC) resides on each disk volume and contains a listing of the names, locations, and attributes of all files on the volume and the libraries to which the files are assigned.

SVS/OS allows file blocks to be cleared at allocation or deallocation (see *VS Enhanced System Access Controls (FSAC) Guide* [36]). The SA can specify that file blocks be cleared by the system whenever a user creates a file, or when additional extents are required for an existing file. The system clears file blocks by overwriting them with binary zeros. The SA can specify that file blocks be cleared by the system whenever a user deletes a file. When a file is deleted, the system overwrites, with binary zeros, the blocks that the file occupied.

Before a disk volume can be accessed by any tasks, it must be mounted. SVS/OS provides three options for mounting a volume: labelled, non-labelled, and bypass-label. Unprivileged users are restricted to accessing a volume mounted labelled, since the other two options allow physical I/O directly to the volume.

For additional information about disk volumes, see "Disk Volumes," page 45.

Data Organization on Tape Volumes

A magnetic tape supports only sequential file organization. A tape volume may be labeled electronically. Such a volume label contains a volume name, and header and trailer information for each file. A tape volume does not contain a VTOC. For additional information about tape volumes, see "Tape Volumes," page 46.

SVS/OS provides two techniques for locating a file on tape: file name or file sequence number. If the tape is labeled, and a file name is specified, then the tape is searched sequentially in an attempt to find the desired file. If a file sequence number is specified, the system attempts to locate the file via its ordinal position on the tape volume (i.e., the file sequence number of the first file on tape is 1, the second is 2, etc.).

2.2.11 I/O Management

SVS/OS supports two forms of I/O to devices: serial and queued. *Serial I/O* is the normal mode of operation, where the system sends one IOCW at a time to the controller and waits for an explicit acknowledgement before performing other activities. The controller must receive a response from the device before processing another I/O request. IOSWs are also handled one at a time.

In *queued I/O*, IOCWs and IOSWs are placed on queue chains. The IOC then accesses the queues directly from memory. This mechanism supports a series of queues.

Each IOC has one I/O Command Queue (IOCQ), which contains entries for I/O operations initiated by a program performing an **SIO**, **HIO**, or **CIO** request. Whenever the system places an entry on an empty IOCQ, the system notifies the controller via an interrupt. Whenever the controller clears an entry from a full queue, it notifies the system of this event with an interrupt. The IOCQ is a circular buffer whose size varies based on the associated controller. Each controller processes the entries on its queue and passes the request on to the destination device. The device responds with an IOSW to the controller which places an entry on one of the I/O Response Queues (IORQ).

The system maintains four IORQs, all of which are available to each controller. Whenever a controller places an entry on one of the IORQs, it notifies the system with an interrupt.

There is also an I/O Free Queue (IOFQ), which is used to supply the controller with IORQ entries for unsolicited I/O events.

Meter Maid (RITA)

RITA is a background dedicated system task which monitors the I/O subsystem for "lost" I/O completions. Every second a counter is incremented in every active Unit Control Block (UCB). The counter for a particular device is reset to zero whenever any interrupt occurs from that device. If the counter exceeds the maximum allowed for that device type and operation, the interrupt is assumed lost. RITA issues an **HIO** instruction and tests the IOC. If it gets a normal response from the IOC it reissues the I/O request. If not, it assumes a problem with the IOC microcode, and attempts to reload the microcode for that IOC.

I/O Interface

SVS/OS's low-level I/O interface available to users consists of the XIO, HALTIO, and LOADCODE SVCs.

The XIO SVC validates disk extents, acquires page frames for input operations if the virtual pages referenced are not in memory, fixes (locks from a page-out) pages for the duration of the I/O operation, constructs the IOCW, builds indirect data address lists for workstation and disk I/O, ensures that the change bit in the page frame table is set for each modified page, and enters the system start I/O routine to initiate the operation.

The HALTIO SVC stops an I/O operation that was initiated by the XIO SVC. If the I/O operation is in progress, the SVC issues the **HIO** instruction to the device. If the I/O is not in progress, the request is removed from the I/O queues.

The LOADCODE SVC loads microcode into a programmable IOP or a device. The system maintains the

name of the library in which all the default microcode files reside. This library name is fixed during system generation (the default name is **@SYSTEM@**). The library resides on the system volume. **LOADCODE** provides to the caller the capability to load microcode to a device or an IOC, or to load a device configuration table to an IOC. Unprivileged callers can load only devices that they have exclusively reserved.

2.2.12 Interprocess Communication

There are two forms of IPC in SVS/OS: *process synchronization* and *interprocess message passing*. Process synchronization can be accomplished through any of three mechanisms provided by SVS/OS: semaphores, user locks, and system locks. SVS/OS also provides three separate mechanisms to support interprocess message passing: ports, mailboxes, and sessions. Details concerning Wang's IPC mechanisms are Wang Proprietary and are not discussed within this document.

2.2.13 The Audit Mechanism

Auditing of security-relevant events in SVS/OS is accomplished by having the software that produces such an event call a process to log the event in a system event log file. An auditable event is a system event that has been predetermined to be security-relevant. All the event types that are considered to be security-relevant, and therefore auditable, are listed below:

- Logons;
- Logoffs;
- Files opened for input;
- Files opened for modification;
- Files closed;
- File renames;
- File deletes;
- File attribute changes (owner, file protection class, expiration date);
- **MSMAP/UNMAP** calls;
- Changes to the file **USERLIST** (see "**VSSECURE**," page 43) and to special program access rights;
- Programs run;
- Procedures run;
- Background jobs run;
- Data processing files printed;
- File references printed;
- Volume mounts;
- Volume dismounts;
- Messages broadcast by operator;
- Messages sent to operator;
- Attach/detach devices;
- Acquire/release workstations;
- Snapshot dumps;
- Task **INVOKES/KILLS**;
- Access control list changes;
- Other changes of **@SECFILE**;

- Operator actions;
- Inter-process communication;
- Reserve/release devices.

An auditable event is recorded for analysis in a log record with a set format for the given type of event. Information pertaining to the subject and object related to the given event, as well as other event-specific information (e.g., access rights for file accesses; date; time; success or failure of the attempted action), is stored in the log record. A complete description of log record formats is given in the TFM (specifically, Appendix B of the *VS Enhanced System Access Controls (ESAC) Guide* [36]).

Audited events are stored in one of the following system log files: the Primary Log File (PLF), the Recovery Log File (RLF), or the Emergency Log File (ELF). All log files are stored on the Primary Log Volume (PLV), or, if the PLV is full, an Alternate Log Volume (ALV). When the logging facility is started, the PLV and log files are provided to it by the caller. If both the PLV and the ALV become full, the logging task will fail and the system will stop running.

When a log file is created, SVS/OS automatically assigns it the Private (#) file protection class (see "File Protection," page 49), and a blank owner ID. This protection insures that only SAs are able to access the log file.

Whenever a task performs an auditable event, it calls the **TESTLOG** macro or **TESTLOGR** system service to see if the event it performed is currently being audited. If so, the task will notify the logging task (**LOGGER**) of the event. This notification (via the **PUTLOG** system service) is accomplished through the datagram mechanism (see "Interprocess Communication," page 36). If **PUTLOG** gets a return code indicating a failure, it will turn on the Event Logging Task Failure Flag and send a failure notification to Workstation 0. Once the failure flag is set, the **PUTLOG** call will not return, and the system will suspend that task. The fact that the attempted audit event failed is recorded in the message sent to Workstation 0. The operator may take action to make more space available to the audit facility. With this mechanism, SVS/OS can guarantee that no more than one audit record per task is lost.

The Logging Task (LOGGER)

The logging task, **LOGGER**, is a dedicated system task responsible for controlling the event logging mechanism. It performs the following functions:

- initializes the event logging mechanisms during IPL or in response to a **CNTROLOG** request;
- processes **PUTLOG** messages;
- handles log file changes;
- shuts down the event logging mechanism in response to a **CNTROLOG** request or logging mechanism failure.

Audit Mechanism Interfaces

The interfaces to the audit mechanism consist of the **LOGPARMS**, **CNTROLOG**, and **PUTLOG** system services, and the **SLFORMAT** and **SLPRINT** utilities. **LOGPARMS** and **CNTROLOG** are privileged services, and an unprivileged

user must specifically be running an application logging event to be able to call **PUTLOG**.

LOGPARMS is used to set security logging parameters and, in the event of a logging failure, to flag **LOGGER** and send a message to Workstation 0.

CNTROLOG allows the SA to select events to be audited, create log files, and specify log file sizes. The selectivity of events is based on three categories: system, file, and user. *System* events are the events auditable for every file and user on the system. *File* events are the auditable events related to a particular file, or set of files. *User* events are the events auditable for a given user ID or set of user IDs, including operators. The **VSSECURE** utility (see "VSSECURE," page 43) enables the SA to select events through a menu-driven interface.

PUTLOG is used to signal **LOGGER** that an auditable event has occurred and to request **LOGGER** to record the event in the current system log file.

SLFORMAT and **SLPRINT** are described in the section detailing the trusted utilities within SVS/OS (see "Trusted Utilities," page 40).

2.2.14 Print Processing

SVS/OS provides two distinct methods for processing printed output: *Data Processing (DP)* printing and *Word Processing (WP)* printing. DP printing produces the standard output from tasks. WP printing handles the printing of documents generated by the Word Processing Application. All print requests are submitted to the TCB through the **SUBMIT** SVC (regardless of the method used to initiate the request). The **SUBMIT** SVC will cause a file to be placed on the print queue for a specific device or print class. The system makes the necessary access checks on the queued file at the time the file is placed on the queue, and generates a log entry for the event.

The print queues reside in system files and are organized by *print class*. A print class is a set of destination printers for print requests. There are 26 print classes (labelled A-Z). These are used for scheduling, load balancing and prioritization of print requests.

The system task (**SYSTSK**) controls all printing and keeps track of the status of all configured printers. At system initialization, a dedicated system task (**PRTTSK**) is created to handle all DP printing. Likewise, a dedicated system task (**WPPRT**) is created to act as a parent task for WP printing. **SYSTSK** communicates with these tasks via privileged ports (see "??," page ??). During system initialization, **WPPRT** communicates with **SYSTSK** to identify all WP-capable printers.

At system initialization, all active printers are acquired by the system and dedicated to DP printing and the appropriate microcode is loaded. When **SYSTSK** detects a DP print request at the front of a printer queue, it sends the request to **PRTTSK** for processing.

When **SYSTSK** detects a WP print request at the head of a printer queue, it releases the printer from the system so that **WPPRT** can acquire it. **SYSTSK** then sends the print request to **WPPRT**, which spawns a WP child task that runs under the user ID of the requestor (the user that queued the file). The child task will control the printing of the file to a specific printer. It acquires the printer, loads the appropriate microcode, and prints the file. **WPPRT** communicates with the child task over an unprivileged port. Note that a WP print file may contain pointers to other files. Access to these files is validated at the time these files are opened by the child task. **SYSTSK** generates the log entries for the start-of-print and end-of-print events.

When printing is completed, the child task notifies **WPPRT**, which then notifies **SYSTSK**. **SYSTSK** then looks at

the next request queued for that printer. If it is a DP request, WPPRT is requested to release the printer. WPPRT notifies its child task to release the printer and terminate. When this is accomplished, SYSTSK will acquire the printer and load the necessary microcode. The print request is then passed to PRTTSK for processing.

If the print request is a WP request, it is passed to WPPRT. If the request is for the same user, WPPRT will pass it to the child task for processing. If the request is from a different user, WPPRT will notify the child task to release the printer and terminate. When this is accomplished, WPPRT will spawn another child under the new user ID and pass the print request to this task.

2.2.15 System Initialization

This section describes software aspects of the bootstrap process and the system initialization process. In addition, it discusses the mechanism for initially loading microcode into system devices.

Initial Program Load (IPL)

After the microcode is downloaded into the IPL device's controller, the file IPLTEXT is loaded from the first 2 KB block on the specified system disk. A branch is then made to the IPLTEXT code.

The first block of IPLTEXT is responsible for loading the rest of IPLTEXT into main memory. After it is loaded, IPLTEXT performs the following processing:

- determines the size of main physical memory;
- relocates IPLTEXT to the top of main memory and places a pointer to IPLDATA in GR15; (IPLDATA is a data area in low memory that contains data used during system initialization);
- finds and downloads microcode to the rest of the microcode-loadable devices (on systems that support microcode-loadable controllers);
- starts execution of the system nucleus file, which includes the following modules: CPINIT, WSOINIT, and VSDUMPER.

At this point, various pointers, PCW fields, I/O tables, UCBs, and the system page tables are created.

When IPLTEXT starts the execution of the system nucleus file, CPINIT is executed first. CPINIT then initializes the system control blocks and sets up the program check and interrupt vectors. CPINIT then puts task control blocks into memory for the following tasks: TSKMGR, PAGER, and RITA. Once these task control blocks have been loaded, CPINIT allocates memory space for the system memory pools, and builds the system tables (e.g., page frame tables, IOCT, SVC vector table). CPINIT's final task is to initialize virtual memory by setting the SCRs and RNTs for the I/O task, Workstation 0 task (WS0), and RITA. At this point, PAGER, WS0, and RITA are capable of being run as tasks, and the scheduler and dispatcher are started. When CPINIT finishes executing, the system starts running under WS0. Control then passes to WSOINIT.

WSOINIT is the second task running under the dispatcher (after RITA and before PAGER). It continues building the operating system structures. In particular, it is responsible for initializing the system disk page pool, initializing the memory manager, and allocating memory for the process-level stack page tables. WSOINIT is also responsible for linking to the @SYSGEN@ module. This module reads the system configuration file,

@CONFIG@, and builds the various I/O structures. WSOINIT then links to @IOSLOAD, which is responsible for downloading the appropriate controllers and devices. When it has finished its work, @IOSLOAD unlinks back to WSOINIT. The final functions performed by WSOINIT are to mount all the volumes, solicit the time of day from WSO, and initialize the dump task, VSDUMPER. Upon completion, WSOINIT renames itself to TSKMGR. All other dedicated system tasks (e.g., LOGGER, PRTTSK) are then created by TSKMGR.

If any fatal errors occur during the IPL process, the system is placed in Control Mode. When the system goes down, the information describing the error and the state of the machine are written to a known memory location.

2.3 Trusted Utilities

This section describes the various utilities included in the TCB.

2.3.1 BACKUP

The BACKUP utility performs backing up, restoration and verification of files, libraries, and volumes. The *backup* function copies original data in files and libraries from a disk to either another disk, or a magnetic tape. The *restore* function copies backed-up data from a magnetic tape or disk to a disk. The *verify* function verifies the backup version of data and creates a list of the files stored on a volume.

BACKUP is considered to be part of the TCB because it must maintain security data (i.e., file protection information). Any SVS/OS user with a valid User ID can run the BACKUP utility, subject to any protection placed on BACKUP. Some functions require the user to have SA or operator privileges (e.g., Bypass Label Processing (BLP) mount requests). If the user of BACKUP does not possess these privileges, it will fail. A user of BACKUP who does not have SA privileges can copy only those files that he or she is authorized to access. BACKUP maintains a log of who is using the utility and the success or failure of access attempts.

2.3.2 BUILDALT and SORT

The BUILDALT utility builds alternate-indexed structures for files containing alternate indexing keys. BUILDALT is invoked as one of the jobs performed by the CLOSE SVC and by any system utility or user program that creates an alternate-indexed file. When BUILDALT is invoked by the CLOSE SVC, it runs at process level 6. BUILDALT calls (via the link-level mechanism) the SORT utility to sort the temporary files it creates; note that BUILDALT's privileges (i.e., process level 6) are also extended to the SORT utility. BUILDALT creates a temporary file, #BUILD, to store the unsorted alternate key values. The SORT utility is called to sort the #BUILD temporary file. It returns the #BUILD file, with the keys in a sorted order, to BUILDALT.

BUILDALT is not directly invokable (i.e., it is not available at the command processor level); it is only accessible through the CLOSE SVC. BUILDALT is considered trusted because it runs at process level 6; the same is true for SORT when it is linked to BUILDALT. The temporary files created by BUILDALT have a default protection class of private.

2.3.3 DISKINIT

The **DISKINIT** utility provides the mechanism to initialize a disk volume. **DISKINIT** provides the following six functions:

- INITIALIZE — initialize a disk volume; all information on the disk is destroyed;
- REFORMAT — provide a *clean* volume label and VTOC; other information on the disk is preserved, but not directly accessible;
- RELABEL — store a new volume label on the disk;
- VERIFY — check accessibility of all disk blocks on the disk;
- REMOVE — remove access to bad blocks on the disk;
- ERASE — erases all data from a disk volume.

DISKINIT can also allocate a dump file or a page pool (or both) on a new disk volume, or change the size of these files on an existing volume. These operations are typically performed when tuning SVS/OS. A *dump file* is a storage area where the contents of main memory can be placed for analysis. This file is used in conjunction with the system dump utility, to analyze snap-shot dumps of the running system. The *page pool* is used to read programs into and out of main memory; it is basically a swap area for temporary storage of main memory pages.

DISKINIT is considered part of the TCB because it is responsible for building the disk structures (e.g., VTOC) that are used as a basis for DAC mechanisms (see "Discretionary Access Control (DAC)," page 48). **DISKINIT** can be run by any user; however, **DISKINIT** performs some functions that require SA or operator privileges (e.g., BLP mounting of volumes). If the user does not possess these privileges, **DISKINIT** will fail. **DISKINIT** can be protected by the standard access control mechanisms to prevent unauthorized access.

2.3.4 FLOPYDUP

The **FLOPYDUP** utility creates a disk image and transfers that image to a disk file or another diskette. A *disk image* is a file containing an exact copy of an entire diskette.

FLOPYDUP is considered part of the TCB because it is responsible for building disk images. **FLOPYDUP** can be run by any user; however, it performs some functions that require SA or operator privileges (e.g., BLP mounting of volumes). If the user does not possess these privileges, **FLOPYDUP** will fail. **FLOPYDUP** can be protected by the standard access control mechanism to prevent unauthorized access.

2.3.5 GROUPEDT

The **GROUPEDT** utility allows the SA to manage groups, which are used in access control of files (see "Subjects," page 44). **GROUPEDT** can be invoked directly from the Command Processor, or can be invoked from the **VSSECURE** utility (see "VSSECURE," page 43) by the SA. It is implemented in Wang's relational database management system, which provides application development tools, including screen definition capabilities.

GROUPEDT is trusted only because it accesses or manipulates the security database files GROUPLST, @DELGRP, and USERLIST. The program itself is accessible by any user, but only the SA will have access to these three files, which are given special file protection (see "Files," page 44).

2.3.6 SLFORMAT

The SLFORMAT utility copies the encoded entries from an event log file into a compressed indexed file and converts them into a format which is more accessible to inquiry and reporting programs. SLFORMAT gives the indexed file it creates the same file protection class, ownership, and ACL as the source event log file.

SLFORMAT is considered part of the TCB because it is responsible for preserving the audit information originally written to the log file when it performs the format conversion. SLFORMAT can be run by any user, but only SAs have access to event log files.

2.3.7 SLPRINT

The SLPRINT utility operates on SLFORMAT-created indexed files to produce report files, in a human-readable format, from event log files. SLPRINT gives the print file it creates the same file protection class, ownership, and ACL as the SLFORMAT-created file on which it operates.

SLPRINT is considered part of the TCB because it is responsible for preserving the audit information contained in the SLFORMAT-created file when it produces its human-readable report. SLPRINT can be run by any user, but only SAs have access to the files created by SLFORMAT.

2.3.8 SYMBOLIC DEBUGGER System Services

The SYMBOLIC DEBUGGER is a collection of utilities and system services that allow a user to locate and correct programming errors in a program. The user interface to the debug environment is @DEBUG@, a program that is initiated as a child task of the program that is being debugged. The debug environment can be initiated by the user by exercising the DEBUG Command Processor option after pressing the HELP key, or after an executing program encounters an error (see "Task Execution," page 25). Mailboxes are created for both the parent and the child task (@DEBUG@), and both tasks are marked in their extended task control blocks to indicate that the user task is being debugged by @DEBUG@.

During debugging, the user task and the debugging task never run simultaneously. They run alternately, as needed, to perform debugging functions. For example, the user program may run until it reaches a specified breakpoint. At this point, control transfers to the child task, @DEBUG@, which analyzes the breakpoint, and returns information to the workstation screen. The transfer of control between parent and child is controlled by messages passed between them through the mailboxes.

Because the debugging task operates in its own address space, it relies on system services to provide it with access to the address space of the program being debugged. These services are provided by PMEMPEEK, PMEMPOKE, PSTAPEEK, PSTAPOKE, PGETTRAP, PSETTRAP, and PCONTROL. These services are described in the *VS System Development Tools* manual [41]. These services, which are the only portions of the SYMBOLIC DEBUGGER that are part of the TCB, provide for the examination and modification of memory, registers, the PCW, and the screen image, and also provide for the setting and examination of breakpoints. The security

checks on these services include a check that the program file is readable, that the invoker is the @DEBUG@ task that is the child task whose memory is being accessed, and that the user task has sufficient access privilege to the memory locations being read or modified. Additional checks are made to prevent a user from examining or setting breakpoints in code other than that running at process level 0.

2.3.9 TAPEINIT

The TAPEINIT utility provides the mechanism to initialize a tape volume. At the user's option, TAPEINIT can erase all the data on the volume, or just replace the volume label and tape mark. TAPEINIT initializes volumes with either the standard SVS/OS or IBM tape label.

TAPEINIT is considered part of the TCB because it is responsible for initializing tapes correctly. TAPEINIT can be run by any user; however, it performs some functions that require SA or operator privileges (e.g., BLP mounting of volumes). If the user does not possess these privileges, TAPEINIT will fail. TAPEINIT can be protected by the standard access control mechanism to prevent unauthorized access.

2.3.10 VOLCOPY

The VOLCOPY utility provides a simple mechanism for performing backups and restoration of an entire disk or tape volume. VOLCOPY performs a byte-to-byte copy of a disk to a disk, a disk to a tape, or a tape to a disk.

VOLCOPY is considered part of the TCB because it is responsible for correctly copying information from one volume to another. VOLCOPY can be run by any user; however, VOLCOPY performs some functions that require SA or operator privileges (e.g., BLP mounting of volumes). If the user does not possess these privileges, VOLCOPY will fail. VOLCOPY can be protected by the standard access control mechanism to prevent unauthorized access.

2.3.11 VSSECURE

The VSSECURE utility manages the system security files. It provides a screen interface that enables SAs to store and maintain data which defines the system access controls, file access controls, and event logging parameters for an installation. VSSECURE requires that the user has SA privilege since several system services are used to manipulate the security database files USERLIST, @SECFILE, GROUPLST, @DELUSER and @DELGRP. In the released system, all of these files have: a file protection class of private; no owner; and an expiration date of 31 December 1999 (see "Files," page 44). They all reside in the library @SYSTEM@, on the system volume.

The USERLIST file contains the User ID, name and access rights of each system user. The @SECFILE file contains global system security options, logon restrictions, default values for a new User ID, and audit control data. The GROUPLST file contains information relating User IDs to Group IDs. The @DELUSER file stores information on users that have been deleted from the system, and is used to prevent inadvertent reuse of such IDs. The @DELGRP file stores information on groups that have been deleted from the system, and is used to prevent inadvertent reuse of these IDs.

2.4 TCB Protected Resources

2.4.1 Subjects

The only type of subject defined in the SVS/OS is a task. A *task* is a process that acts on behalf of a specific user. See "Task Execution," page 25 for a detailed discussion of tasks. Once the user has been authenticated, the attributes of that user become the static attributes associated with the task. The static attributes, which are specified for each user by the SA, consist of the following: a user ID, group membership, role, and file protection class access rights.

In addition to the static task attributes, each task has the dynamic attributes of privilege and Special Program Access Rights (SPARs). SPARs are comparable to file protection class access rights, but they are associated with a program, not a user (see "File Protection," page 49).

2.4.2 Objects

Wang defines the system objects as files, disk and tape volumes, tasks, printers, workstations, I/O controllers, and VTOCs and queues. Of these, the named objects subject to access control are files, tasks, and workstations. The remaining objects are internal to the systems (i.e., they can be accessed only by the SA or operators). Access to files is controlled by ACLs, and access to the remaining objects is controlled by other appropriate methods.

Files

The attributes that can be associated with a disk file are: the user ID of the owner; the file type and organization; an expiration date; membership in a file protection class; and an access control list.

The user ID associated with a file identifies the owner of the file. An unprivileged user has unlimited access to all files owned by that user. Further details about user IDs are provided in "Logon ID," page 47.

The expiration date specifies a minimum retention period for the file. The command processor, or a procedure run from it, cannot delete or rename a file before the specified expiration date.

The different file types and organizations, which determine to some extent the operations that can be performed on a file, are described in "File System," page 33.

File protection classes and access control lists are fully described in "File Protection," page 49.

Certain critical system files (`USERLIST`, `GROUPLST`, `@SECFILE`, `@DELUSER`, and `@DELGRP`) are controlled so that only the SA can access them. The controls on these files are: private, no owner (meaning that no one has owner access), and an expiration date of 31 December 1999.

Tasks

Tasks communicate with each other through the use of interprocess data structures (i.e., mailboxes, ports, and sessions). While they may be viewed as objects, interprocess data structures are not considered named

objects (and, therefore, not subject to the same access control restrictions as named objects) because they are simply the interfaces to the tasks they address. Each is created for a particular task, and is destroyed when that task finishes. Because it is the task that is being referenced through these objects, the task is considered to be the named object.

Workstations

An SA or operator can determine if a specific workstation is available for use. The SA or operator may also force a user to logoff. A workstation that is detached cannot be used by any task. A workstation that is attached but released can be acquired by any task. The system acquires workstations at system initialization and then assigns a workstation to the user task during the logon process. Once acquired by a task, the task has exclusive use of the workstation. A workstation must be acquired before user logins are permitted. The attributes associated with a workstation are: whether logins are prohibited and whether automatic logoffs are supported.

Disk Volumes

The dynamic attributes associated with a disk volume, which are assigned when the disk is mounted, are: bypass label processing (BLP), access type, spool/work/page, and the disk security status *BLP mounting* allows any block on the disk to be read or written by a user authorized to mount the disk. Because all disks are owned by the SA or the operator, a request to perform a BLP mount of a disk can come only from an SA or an operator. (Such requests are mediated by the TCB.)

The *access type* is one of four states specified on a disk mount request: exclusive, shared, restricted removal, and protected. The *exclusive* attribute specifies that only by the user who mounted the volume can read from, write to, or dismount the disk. The *shared* attribute indicates that any user can read from, write to, or dismount the volume. Access to specific files on the disk is subject to the discretionary controls on that file. The *restricted removal* attribute indicates that any user can read to or write from the disk, but only the user who mounted the disk can dismount the disk. The *protected* attribute specifies that any user can read any file subject to the file's DAC, but writing to files or dismounts of the disk can only be accomplished by the person who mounted the disk.

The *spool*, *work*, and *page* attributes indicate whether the disk accepts spool, work, and/or page files. The spool or page attribute is allowed only on mounts of a labelled disk requested by the SA or an operator. The spool and work attributes indicate that spool and work files may be placed on this disk, for users who don't have a default spool or work volume specified.

The *security status* specifies whether SPARs are present for programs that reside on the disk. Only an SA can request the mount of a disk volume such that SPARs are present.

A disk volume can have a Wang-defined label or it is considered to be non-labelled. When a non-labelled volume is mounted, the entire volume can be read or overwritten. Requests for the mount of a non-labelled volume (NL mount) is mediated by the TCB and can be performed only by an SA or an operator. A labelled volume cannot be mounted with an NL request.

Auto-mounting is a capability of mounting a volume without an explicit mount request from the system; when the disk is spun up, the system automatically mounts it. The volume is recognized by the system

as being mounted in shared mode if it is a labelled volume. Auto-mounting of non-labelled volumes is prohibited.

Tape Volumes

Tape volumes are used to import and export data from the system and for backup. The attributes of a tape volume are its type and how it was mounted. Tape volumes are either scratch or private. *Scratch volumes* are blank (erased) and are identified by having either a unique Volume Protection Label (VPL) or no label at all. *Private volumes* are identified by a unique physically-attached volume label. The operator ensures that a tape has been completely erased before its allocation as a private tape volume. Tape volumes are owned by the system and only the system can read from or write to tape volumes; untrusted users have no access to them. Only SAs or operators can mount tape volumes.

Printers

Printers can exist in three states which can be specified only by an operator.

- *Detached* — This state implies that the printer can not be used by any task.
- *Attached but Released* — This state implies that a printer can be opened for "on-line" printing by any task. Once opened that task has exclusive use of the printer until it is released. This state is used primarily by the *WP* facility (see "Print Processing," page 38).
- *Attached and Acquired* — This state implies that this printer can only be used for *DP* print spooling for those print classes assigned by the operator. There are no restrictions on what print classes can be used by a given task.

I/O Controllers

The attributes associated with an IOC are: whether it is microcode-loadable, the file from which the microcode is loaded, and the devices that are attached to that controller. Operators or tasks with diagnostic privilege can load microcode. The `DEVLIST` file specifies the correct microcode file for the controllers. Device access is determined by the task's access rights to the IOC that controls that device.

VTOCs and Queues

VTOCs and queues are objects internal to the TCB. While untrusted tasks can indirectly influence the contents of VTOCs and queues, the modificaction is nevertheless performed only by trusted tasks. The two forms of queues that are maintained by the system in system owned files are print queues and the procedure queue. Like VTOC entries, any queue entry can be read by any task. Only the SA, operator, or the task that submitted an entry can modify or delete that entry.

2.5 SVS/OS Protection Mechanisms

SVS/OS employs three basic mechanisms to protect the resources identified in previous section: Identification and Authentication (I&A), Discretionary Access Control (DAC), and Object Reuse.

2.5.1 Identification and Authentication

The I&A mechanism consists of three components: the logon ID; the user ID; and the password. I&A data are stored in the **USERLIST** system file which resides in the library **@SYSTEM@** on the system volume. The SA can protect the **USERLIST** file from access by unauthorized users, and it is recommended that the **USERLIST** file be protected by encryption.

Logon ID

Users must identify and authenticate themselves to SVS/OS before they can be associated with any given task. The logon ID is a unique identifier assigned to each user by the SA. The SA determines who can use the system, how users must logon, and what resources users can access. SVS/OS supplies special security features which allow the SA to identify users to the system, validate the identity of the user, and restrict access after invalid login attempts.

The logon ID is a unique string (1 to 8 characters in length) assigned by the SA specifically for the purpose of identification, and is requested by the system at logon. SVS/OS provides logon restrictions and options based on the logon ID that are controlled by the SA which include:

- An access restriction after invalid logon attempts that occur within a specified period of time;
- A logon ID lock which prohibits users from logging on to the system if their logon ID is unused for a specified number of days;
- A restriction on logons specified by time and date;
- An enforcement of minimum logon ID lengths (1 to 8 characters);
- A notification to the user of the last date and time of a successful logon;

User ID

The user ID is a three-character unique code, assigned by the SA, which identifies the user to the system and is mapped from the logon ID after a user successfully logs on. If the SA fails to assign a logon ID for a user, the system automatically assigns the user ID to serve as the logon ID. The SA and the system employ user IDs to make all access decisions; logon IDs are not used for this purpose. During a successful logon, SVS/OS via the user ID determines what functions the user can perform on the system, what access rights the user has to file protection classes, and what privileges the user has been granted.

The SA may define groups of users, which are lists of user IDs collectively identified by a group ID. Group IDs are treated the same as user IDs and have the same access modes associated with them. SVS/OS allows

users to belong to more than one group but does not provide a mechanism for them to review the groups to which they belong. The SA must provide group IDs to file owners via site specific administrative methods and identify group members so that file owners can make intelligent access decisions.

Password

A user is authenticated to the system by supplying a password (1 to 8 characters in length)[15] at logon time. SVS/OS provides the following additional password management features to the SA to further enhance protection against unauthorized access to the system:

- Forced minimum password length;
- A system-enforced password expiration date;
- A mechanism to prevent a user from changing the assigned password;
- Forced random password generation;
- An option for passwords to be encrypted;
- A password history file so that passwords are not reused.

2.5.2 Discretionary Access Control (DAC)

Access modes are hierarchical and specify the way a user (or group) can access protected files. An access mode may be one of the following:

- Execute access is meaningful only for program files. Files in classes for which this level is specified can be run by the user but cannot be read directly (i.e. they cannot be copied or linked into another program via the linker, nor can they be examined or modified via the debug processor);
- Read access provides read and execute rights; files cannot be modified. Program files can be copied, linked, and debugged, and data files can be opened in INPUT mode only;
- Write is the highest level of access and includes Read and Execute access as well as the ability to modify, rename, and delete files;
- Null access (denoted by a blank) explicitly denies access.

Upon the creation of a user, the SA defines the access modes that a user is to have. Once logged in and associated with a task, the user initially inherits these SA-defined access modes.

[15] The minimum password length suggested in the TFM for the C2 system is 8.

File Protection

SVS/OS provides two file protection mechanisms: the File Protection Class and the Access Control List (ACL).

A file protection class is a mechanism which allows every file that a user creates to be associated with one of 30 available file protection classes. Four are defined by the system and 26 are defined by the SA. Each file protection class is represented with a one character code; @, #, $, " ", or one of he letters A through Z. The SA defines the classes A through Z by specifying which users have access to each file protection class letter. Upon initiation of a new user, the system administrator must define the access (R, W, E, NULL) the user is to have to each of the 26 available file classes via the **VSSECURE** utility.

Upon file creation, the creator assigns the file a protection class. Users with access rights to the file's assigned protection class may then access the file. If a file protection class is not specified, the task default (if set by the owner of the task) is assigned; otherwise the system default is assigned.

The system defined file protection classes available to users are " ", @, $, and #.

- " " - denotes an unprotected (public) file; every user has W access;
- @ - denotes execute only; every user has execute access;
- $ - denotes read only; every user has read access;
- # - denotes private; the owner has write access; all other users have null access.[16]

An ACL is an optional file protection mechanism that one may use in addition to file protection classes. ACLs provide the user with a more specific and flexible method of file protection, allowing file owners to share their files by specifying on the list the user/group IDs and accesses those IDs are to have. If a file has an ACL, the ACL is maintained in the VTOC associated with the file. An owner can create a unique ACL for each file that is viewable only by those with read access to the file. Only the owner or SA can modify or remove an ACL associated with a file. SVS/OS, through **VSSECURE**, allows up to 45 users and 30 groups to be listed within an ACL, for a maximum of 75 entries per ACL. The format for an ACL entry is as follows:

((Type,Name,Access-Mode)...(Type,Name,Access-Mode)...)

where

- Type specifies either U (individual user) or G (group);
- Name specifies either the user ID or the group ID;
- Access-level is W (write); R (read); E (execute); or blank (null access).

Programs (executable code) can be given access to files. The SA can give a program special access rights (SPARS) of its own which are independent of those possessed by a user running the program. These programs that have been granted SPARS assume both their own and the owner's access rights while the program is running. The SPARS given to a program are not passed to the user running the program. These

[16] C2 file class default.

SPARS can be either SA access rights (i.e. write access to all files) or access rights to a subset of the file protection classes.

When a user (or program) requests access to a file, a number of checks are made before the user is granted access to the file. SVS/OS uses the following decision hierarchy of checks to determine a user's eligibility for file access:

1. Is the user the owner of the file, the system administrator, or running a program with an SA SPAR? If so, the user is granted write access.
2. Is the user explicitly defined in the file's ACL? If so, the user is granted the access specified in the ACL.
3. Is the user a member of a group that is defined in the ACL? If so, the user is granted the access associated with that group entry in the ACL.
4. Is the user running a program that has SPARs allowing access to the file? If so, access is granted based on the union of the program's SPARs and the user's accesses.
5. Does the user have access to the file protection class of this file? If so, the user is granted the access that the SA has given him to that file protection class.

Specific examples of file access checks are given within the *VS Security Features User's Guide (SFUG)* [33].

2.5.3 Tasks

Tasks may communicate with other tasks via the Interprocess Communication Mechanisms which Wang has defined as ports, mailboxes and sessions. To communicate with another task, a task must first create one of these interfaces to which another task will send data.

For ports, read and delete access is granted only to its owner. If the port attribute specifies that only privileged tasks can append messages, then append-only access is granted to tasks in rings one through seven or with PCW bit 34 set to indicate privilege; otherwise the ability to write to a port is denied. Like ports, mailboxes grant read and delete access only to their creator. Append-only access is granted to any task at or above a the ring level specified by the creator. Sessions provide a point to point communiciation mechanism which identifies a single sender and receiver. Session access, if granted to a task, permits access to a specific session, for use in subsequent message exchange with another task. Each task is granted read and delete access to its own mailbox, and append-only access to the mailbox of the other task in the session. A task not identified as a sender or receiver may not communicate via this mechanism.

A task chooses which one of these IPC mechanisms to use based on the needed functionality and granularity of control required by the executing application.

Tapes

All tapes are owned by the system and can only be accessed by the SA or operators.

Printers

All printers are initially owned by the system unless they are specifically released or detached by the TCB or the operator. Each printer has associated with it a print class which corresponds to the submitted print class of the files which it may print. This print class is not related to the protection class of files. When a user submits a print job the print class to be associated with that job must be defined. The user has full control of what print class is to be selected and is not constrained by the protection class of the file when making the print class selection.

Workstations

All workstations are owned by the system unless the operator explicitly releases them. Once a workstation is claimed by a user through a logon, that user (and the associated task) has exclusive use of that workstation. Upon release, the system resumes ownership of the workstation and makes it available to other users for logon. If the operator releases a workstation from system ownership it may be acquired directly by any task.

IOCs

IOC access is determined by the user's access rights to the specific devices controlled by the IOC. Only operators and users with diagnostic privilege are permitted to load microcode to IOCs; otherwise, when initialzed, code from the Device List (which is maintained by the SA) is loaded.

Queue Entries

See "VTOCs and Queues," page 46.

2.5.4 Object Reuse

Information held in the following objects is protected by clearing them before they are allocated to different users:

Tasks: A user task is invoked by the task manager when the login process is completed for a workstation. This user task initially executes in the command processor before it executes a user program. The initial register contents available to a user program consist of the residue from the TCB code initially executed by that task. All memory areas allocated to the task are cleared before allocation.

On a task context switch in the CP, the TCB saves the entire process state, to include the general registers, pertinent control registers and floating point registers. This entire state is reestablished when this task is again put in control of a CP. In addition, all cache is cleared on every context switch.

When a virtual page is deallocated by a process, it is released to a pool of available pages. A "no-info" bit is set when this page is unlinked from a program, indicating that the information in the paging file associated with this page is not valid. When the pager requests this page for another process, it recognizes (via the

no-info bit) that the page contains no useful information. Instead of reading this page into the cache and clearing it, the page in the cache is cleared without doing the read. When the page in the cache is written back to virtual memory, the information contained in the previous virtual page is overwritten.

An initiator task is a task that gets reused for user background tasks. The initiator task, running TCB code, clears itself before executing on behalf of another user.

Files: File space reuse is specified by the SA through the use of a system configuration parameter. The SA can specify file space scrubbing on either release of the space, on allocation of the space, or both. The default setting for the evaluated system is to clear on release. Since the only space on disk volumes that is available to an unprivileged user is file space, there is no object reuse issue with disk volumes.

Tape Volumes: Tape volume reuse is handled through administrative procedures. These procedures are described in the appendix to the *Security Features Users Guide (SFUG)* [33]. This reference instructs the operator to run TAPEINIT when a tape is changed from public or private to scratch or from to scratch to public or private.

Mailboxes/Ports/Sessions: Mailboxes/Ports/Sessions consist of control structures and messages that are not directly accessible to a user. The control structures are cleared before being used for another user. The message buffers are not accessed by a user but are copied from the message buffers to user space. The system maintains the message length and transfers only the message into the user buffer.

Printers: Printers are loaded each time the device status is changed to "released" or the ownership of the device is changed. At this time the device is cleared and reinitialized.

Queue Entries: All system queues (i.e., Procedure/Program Queue and Print Queues) reside in a single file, @QUEUE@, and are write-protected from an unprivileged task. Since the information in all queues can be viewed by any user, there is no object reuse issue related to the queue entries themselves.

Workstations: During the logon process, a workstation is acquired by a user task. Before the workstation is actually allocated to the task, workstation micro-code is reloaded onto the workstation to overwrite any remnant micro-code or buffer data that the previous owner of the workstation may have left.

Chapter 3

Rating Maintenance Phase

Since Wang is participating in the Ratings Maintenance Phase (RAMP) for SVS/OS, Wang has a configuration management plan in place. Wang maintains control of changes to SVS/OS source code, test code, documentation, and hardware. The SVS/OS Rating Maintenance Plan (RM-Plan) [1] explains the configuration management procedures.

Wang's SVS/OS RM-Plan is managed by the leader of the Security Development Group. This person designates the lead Vendor Security Analyst (VSA) and other VSAs; establishes VSA responsibilities; communicates with the NCSC on administrative and programmatic issues; and provides corporate assurance that the RM-Plan and submissions of evidence accurately conform to the RAMP process.

A Wang lead VSA is responsible for the execution of all technical tasks in RAMP, including the presentation and defense of SVS/OS evidence. This VSA is a member of Wang's Security Development Group which oversees all RAMP activities and provides technical support to the VSA.

Changes to SVS/OS TCB software modules are categorized either as projects or bug fixes. Each project and bug fix is individually analyzed and assessed for its relevance and impact with regards to protection from unauthorized disclosure, modification, and delay and denial of service. All projects (i.e., all changes to SVS/OS) and bug fixes are made to the system via formal builds, with each build representing a formal internal checkpoint which is documented by a Software Release Notice (SRN).

Each proposed project must have Functional and Design Specifications or a Technical Specification, and each specification must be approved by a set of "required reviewers," one of whom is the lead VSA. In order to be considered for development, a Project Request must be presented to Wang's Host Systems Development Strategy Review Board (SRB). If the SRB approves the request, the Host Systems Development Group is asked to assess the feasibility of implementation, estimate the scope of the project, and generate an architectural sketch outlining an approach for design and implementation. The architectural sketch is then presented to the Architectural Review Board (ARB) for approval and acceptance. If accepted, a Host Systems Project will be initiated to satisfy the request and assigned to a future release of the SVS/OS, using Wang's Monitor Change Order (MCO) process. This process is discussed within a later paragraph. The SRB and ARB are the corresponding entities in the Wang SVS/OS configuration control process that correspond to the Configuration Control Board (CCB) described within [1].

A Wang Security Reviewer and the lead VSA reviews the symptom, diagnosis, and cure for each bug fix and determines if the fix is security relevant. If it is, the applicable MCO is immediately annotated.

Wang's MCO process governs, regulates, and tracks all changes, including projects and bug fixes, to the software, macros, and procedures of the SVS/OS, ESAC, CAP, utilities, XDMS, and Device Packages. An MCO is equivalent to the Engineering Change Order (ECO) which is described within [1]. Wang has defined each SVS/OS TCB software module as a configuration item (CI). Each CI is maintained within Wang's MCO database, where Wang's database management system (PACE) can generate any necessary reports on demand. Information kept within the MCO database includes MCO number, MCO author, code reviewer, approval signatures, symptom, diagnosis, cure, test plan description, files modified and their location, the

checkpoint the MCO was released in, and various date and status information.

Security test plans and associated code for projects and bug fixes which are security relevant are written and managed by Wang's System Integration and Test (SIT) organization. These plans and related code are maintained both electronically and manually. The lead VSA is responsible for monitoring all changes in security test plans and related code.

Wang uses a manual system to track changes to documentation. Each document is uniquely numbered, and all changes, literary and technical, are indicated with revision marks. Each document has a log which indicates the date a change was made, the page(s), and whether the change was security relevant or not. The lead VSA is responsible for monitoring all documentation changes.

The underlying hardware is not under configuration control within Wang's Security Development Group. NCSC's RAMP document [2] requires that the hardware be configuration identified and analyzed. Wang's tracking process requires that the lead VSA examine the design and testing documentation of new hardware when it becomes available. In effect, the lead VSA performs a complete mini-evaluation on the hardware. The lead VSA documents this analysis, keeps it on record, and presents the analysis to the Security Development Group.

If changes made to the underlying hardware are deemed potentially harmful to security by the lead VSA and the Security Development Group, that hardware will not be approved as an acceptable base, and SVS/OS will not be ported to that base.

Chapter 4

Evaluation as a C2 System

4.1 Discretionary Access Control

Requirement

The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Applicable Features

SVS/OS controls access between named users and named objects. For disk files, access between named users and named objects (files and programs) can be mediated. There are two mechanisms that allow SVS/OS to meet this DAC requirement. These are the use of an ACL and the use of a file protection class. For more detailed information see "File Protection," page 49.

Some SVS/OS disk file types (e.g., work and temp files) are, by default, given the private file protection class ("#"), and thus do not require an ACL since they are owner-accessible files. These files are subsequently destroyed upon termination of the application program or logon session.

SVS/OS allows for a default file protection class to be assigned when a file is created. The default file protection class for system-created files is "#." The default file protection for users is user settable, but is "#" by default. Users can also specify a default ACL to be used when creating files.

Ports and sessions also support a DAC mechanism for the underlying task.

Access to tape volumes is limited to the system, and only the system can read from or write to tape volumes.

Other system objects include devices and queues. For information about access controls associated with devices and queues see "Printers," page 46 and "VTOCs and Queues," page 46.

Conclusion

SVS/OS satisfies the C2 Discretionary Access Control requirement.

4.2 Object Reuse

Requirement

All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Applicable Features

SVS/OS objects are cleared before they are allocated. These objects include task state; files; tape volumes; mailboxes/ports/sessions; devices; and queue entries. For specific information about each of these objects and how they are cleared, refer to "Object Reuse," page 51.

Conclusion

SVS/OS satisfies the C2 Object Reuse requirement.

4.3 Identification and Authentication

Requirement

The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Applicable Features

SVS/OS CAP 1.0 requires users to logon before they can perform any operations on the system. The logon process requires the user to provide a logon ID and a password. The logon ID maps onto a 3-character user ID which is assigned by the SA, and which uniquely identifies the user to the system. For the system being evaluated, the suggested logon ID length is 8-characters, and the suggested minimum password length is 8-characters.

SVS/OS CAP 1.0 supports event logging and allows the SA to create, maintain, and protect an audit trail on specified events such as logons and file access. The SA is able to select event logging activities at the individual user level, with the audit record containing information such as the user ID, date/time stamp, and the event identifier.

For more detailed information about identification and authentication see "Identification and Authentication," page 47.

Conclusion

SVS/OS satisfies the C2 Identification and Authentication requirement.

4.4 Audit

Requirement

The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

Applicable Features

SVS/OS provides a mechanism to record information about the occurrence of security-relevant events. The information is recorded when the event occurs. The recording is accomplished by the inclusion of a call to the audit collection facility in the code that produces the event.

SVS/OS provides the capability of selecting the types of events to be audited. The administrative interface to the audit mechanism is through **VSSECURE**, which in turn uses the **CNTROLOG** system service. This facility provides the functionality of selection of audit events, creation of audit data files, and setting of audit data file sizes.

The SA has the capability to selectively audit the actions of an individual user or collections of users based on the User ID.

The audit files are owned by the SA, and only the SA can write to or read from them.

SVS/OS provides the capability of viewing the recorded audit data with the **SLPRINT** or **SLFORMAT** system utilities. It allows the SA to view the audit data collected into the specified audit data files. The audit data files, or log files, are protected from unauthorized access by the DAC mechanism. Loss of audit data is prevented by halting activity if the log files are full.

For more detailed information about auditable events see "The Audit Mechanism," page 36.

Conclusion

SVS/OS satisfies the C2 Audit requirement.

4.5 System Architecture

Requirement

The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

Applicable Features

SVS/OS isolates the TCB from users by using the hardware process level mechanism. Users run in process level zero, while the TCB runs in process levels one through seven[1].

However, when bit-34 of the PCW is cleared, the task has the ability to access information in any process level. This bit can be cleared only by a privileged instruction, therefore only a task with the bit already cleared or a task executing in process level seven can clear it. It is also the case that when any SVC is invoked, bit-34 of the PCW is cleared automatically. JSCI routines (which execute with the caller's context) do not modify bit-34 of the PCW. SVC routines (which save and restore the caller's context) do not modify bit-34 of the caller's PCW which has been saved on the stack[2].

TCB memory data structures are protected by hardware memory protection based on the process level. All files owned by the TCB are protected by the system's DAC mechanism. Critical system files have the following attributes: private with no specified owner, and an expiration date of 31 December 1999. Access to the TCB is through a set of well defined instructions (e.g., SVCs and JSCIs).

SVS/OS resources are described in "TCB Protected Resources," page 44 of this report. These resources are all subject to the TCB access controls and audit requirements.

Conclusion

SVS/OS satisfies the C2 System Architecture requirement.

[1] There are some dedicated system tasks that run in process level zero and are part of the TCB.

[2] There is one exception to this rule, the PLEASE SVC allows an unprivileged task, which was once privileged, to regain its privilege. In order to do so, the task, while privileged, must have saved a return address in its extended task control block which will be used by PLEASE when restoring the privilege. This cannot be done by an unprivileged task.

4.6 System Integrity

Requirement

Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Applicable Features

Wang provides power-up hardware integrity tests to ensure the correct operation of the various system components. These tests provide the system with an automatic way of testing the status of the hardware components before the operating system takes control. If any of these tests fail, the system will not be allowed to boot until the problem has been addressed.

Wang supplies a diagnostic package for each CP-type. The *Wang Trusted Facility Manual (TFM)* [3] instructs the SA on the procedure to be followed should the SA desire to have diagnostic tests run, and contains references to specific diagnostic documentation.

Conclusion

SVS/OS satisfies the C2 System Integrity requirement.

4.7 Security Testing

Requirement

The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

Applicable Features

Overview of Vendor Test Suite

The complete Wang SVS/OS test suite consists of three distinct parts: the security test suite, the functional tests, and the Wang Workstation test suite. Although there were several instances where security tests and functional tests overlap, Wang felt it was important to separate the two, and perform each independent of the other.

The security tests are aimed at ensuring that all C2 security requirements as stated in the TCSEC are met. The vendor test suite covers discretionary access control, object reuse, identification and authentication, and audit.

Team Augmentation of Vendor Coverage

In addition to Wang's security and functional tests, the team developed several of its own tests, which included:

- Testing that Object Reuse is implemented on Wang Workstations;
- Testing the limits of the system (e.g., forcing overflow of the audit log);
- Testing for Trap Doors (e.g., involving the SCU);
- Testing all privileged SVCs.

Performing Team Testing

The evaluation team conducted testing on Wang VS7310 (CP8), VS7150 (*CP8*), VS85 (*CP4*), VS75E (*CP7*), and VS5000T (*CP9*) systems. The systems were owned by the vendor and located at the vendor site in Lowell, MA. The evaluation team executed all of Wang's automated and manual tests on the *CP8* systems and executed several of these tests on the other CP-types. The tests were run using Release **7.33.36**.

The hardware configuration of computers on which the tests were run is as follows:

VS Series	Description
VS7310 and VS7150 (CP8)	(2) System Console Units (2) Archiving Workstations w/Floppys (8) Workstations (1) Color Workstation (1) 1100 LPM Band Printer (1) 600 LPM Band Printer (2) 8 PPM Laser Printer (1) 800/1600 BPI MAG Tape (1) 800/1600/6250 BPI MAG Tape (1) Streaming Cartridge Tape (2) 75 MB REM SMD Disk (2) 288 MB REM SMD Disk
VS85 (CP4)	(1) Archiving Workstation w/Floppy (3) Workstations (1) 8 PPM Laser Printer (1) 800/1600 BPI MAG Tape (1) Streaming Cartridge Tape (1) 75 MB REM Winchester Disk

(2) 147 MB FIX SMD Disk

VS75E (CP7)
(1) Archiving Workstation w/Floppy
(4) Workstations
(1) 600 LPM Band Printer
(1) 8 PPM Laser Printer
(1) 12 PPM Laser Printer
(1) 800/1600 BPI MAG Tape
(1) Streaming Cartridge Tape
(1) 1.2 MB 5.25" Floppy
(2) 145 MB FIX Winchester Disks

VS5000T (CP9)
(3) Workstations(T)
(1) 1.2 MB 5.25" Floppy
(1) 326 MB FIX Winchester Disks
(1) 145 MB FIX Winchester Disks
(1) 72 MB FIX Winchester Disks
(1) Streaming Cartridge Tape

Security testing for the Wang Workstation consists of verifing that Object Reuse, implemented in microcode, completely clears local workstation memory and microcode is reloaded in the workstation to prevent any software tampering. No extra testing is performed for Audit or DAC since the Wang Workstation introduces no extra auditable events or discretionary access controls.

Problems Uncovered During SVS/OS CAP 1.0 Testing

The team found numerous small omissions and unclear passages in the Test Plan which created some confusion during setup and execution. These documentation problems have been addressed by the vendor and corrected in the final documentation.

Minor omissions within the TFM, such as how to IPL the VS85 (*CP4*), were discovered during testing. The vendor has corrected these problems.

The team did not find any security relevant design flaws during testing.

Conclusion

SVS/OS satisfies the C2 Security Testing requirement.

4.8 Security Features User's Guide

Requirement

A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

Applicable Features

Wang has written a 3-chapter *VS Security Features User's Guide (SFUG)* [33] which is designed to assist all users on Wang systems supporting Enhanced System Access Controls (ESAC).

The "Preface" of the SFUG contains a listing of Wang manuals which contain additional information on the topics included within the SFUG. In addition, a reading path for first time users appears within the "Preface."

Chapter 1 of the SFUG is entitled "Overview of Security" and discusses security features available to ordinary (unprivileged) users, operators, and the SA.

Chapter 2 of the SFUG is entitled "Logon IDs and Passwords" and describes to users precautions which can be taken to protect logon IDs and passwords. Specific topics include how to change a password, how to select a secure password, and troubleshooting logon problems.

Chapter 3 of the SFUG is entitled "File Protection." Topics discussed include file protection classes and ACLs, system default file protection class, and file access rights. This chapter describes how the operating system checks for access rights when a user attempts to access a file on the system, and provides several examples. A useful table that provides a summary of the interaction of file protection classes and ACLs is also provided, as well as suggestions to the user for protecting output and print files. Finally, a discussion of some of the features that are available for protecting the print and output files created by background jobs is presented.

Appendix A of the SFUG discusses the procedures for tape volume protection.

Conclusion

SVS/OS satisfies the C2 Security Features User's Guide requirement.

4.9 Trusted Facility Manual

Requirement

A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given.

Applicable Features

Wang has developed and identified several documents which comprise their Trusted Facility Manual (TFM). These documents are the *Wang Trusted Facility Manual* [3], the *Secure VS Operating System With Controlled Access Protection Guide* [5], the *Secure VS Operating System With Controlled Access Protection (SVS/OS With CAP): Release 1.00 Software Bulletin* [6], the *Secure VS Operating System With Controlled Access Protection (SVS/OS With CAP) Release 1: Customer Software Release Notice* [7], the *VS Enhanced System Access Controls (ESAC) Guide* [36], the *VS System Operator's Guide* [38], the *VS GENEDIT Utility Reference Update: Release 7 Series* [8], and the *VS Backup Utility Reference* [9].

The TFM set provides the SA with the information necessary to establish and maintain a C2 level of security. This information includes instructions for installing and configuring a C2 system; cautions and guidelines for controlling functions and privileges; and instructions for using the Event Logging Facility. These later instructions include how to maintain the log file, how to examine log files, and descriptions of log file record structures.

In addition to the above, the TFM contains pointers to physical security documentation, and a caution about physically protecting Wang Workstations since they are TCB components.

Further, chapter 1 of the SFUG includes security features usage information specific to all users (including the operator and SA). Specifically, operators are informed about logon/logoff control, control of interactive and non-interactive tasks on the system, print and procedure queue control, printer control, and the control of disk and tape volumes. Specific SA functions are also discussed in this chapter including defining the users of the system, specifying when users can access the system, and specifying which system resources users can access[3].

Conclusion

SVS/OS satisfies the C2 Trusted Facility Manual requirement.

4.10 Test Documentation

Requirement

The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

[3]Specific topics include the Event Logging Facility, password management, logon restrictions, the assignment of SA and operator privileges to users, resource access privileges, file protection classes, special program access rights, groups, clearing of file blocks at allocation or scratch, and restricted mounts for bypass label processing and non-label disks.

Applicable Features

Wang has written the *VS System Security Test Reference* [18], a 9-volume, 51-chapter manual describing its testing philosophy, security test plans, and procedures. The manual was written by the Wang Test Group. This group analyzed the TCB interfaces and developed tests for each interface component determined to be security relevant.

Prior to testing, the evaluation team determined that test plans and procedures were lacking for several trusted utilities and for Wang Workstations. Wang developed test plans to address these areas and the plans were provided to the team prior to team testing.

Conclusion

SVS/OS satisfies the C2 Test Documentation requirement.

4.11 Design Documentation

Requirement

Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

Applicable Features

Wang provided extensive design documentation which included the *VS System Informal Security Policy Statement, Version 2.1* [30], which contains a comprehensive description of the subjects, objects, and protection mechanisms within the system.

The internals of the TCB and its trusted processes are presented and discussed within the *VS Operating System Internals Manual* [39] and the *VS Utilities Internals Manual* [40]. This two-volume documentation set covers the internal design of the TCB, and describes the mechanisms through which the system security features are implemented.

Wang also provided a two-volume set entitled *VS Principles of Operation* [27]. Volume 1 describes the machine architecture of Wang VS systems, and Volume 2 describes device I/O.

Conclusion

SVS/OS satisfies the C2 Design Documentation requirement.

Chapter 5

Evaluator's Comments

The evaluation team found Wang's SVS/OS to be a user-friendly, flexible, general purpose operating system when run in its evaluated configuration. During the course of working with the system the team developed comments about some of the system's design characteristics and features. These comments follow.

- The team found that SVS/OS IPL'd on all evaluated CP-types without major difficulty.
- The SVS/OS TFM consists of several documents, including a pointer document. The team found that due to the overall size of the TFM, it was difficult locating information in some instances.
- The team found the audit controls available to the SA to be exceptional. The audit pre-selection choices allow a very fine grain of auditability. Further, the choice of the initial sizes and disk volumes for primary, recovery, and emergency audit log files allows the SA to insure that no audit data will be lost and that the system will rarely halt as a result of auditing, in which case sufficient warning will have been provided.
- The team noted that SVS/OS is based on a flat file system architecture.
- Wang was very cooperative with the team during team testing and successfully developed several tests at the request of the team.
- The team found that Wang has implemented a very good system for managing changes to their TCB software. Wang RAMP activity for SVS/OS will rely heavily on this configuration management system.

Appendix A

Evaluated Hardware Components

A.1 CPU Types

The following list describes all of the CPs (by type, series, and model number) included in the C2 evaluated configuration.

CP Type	VS Series	Model Number
CP12	*VS 8200*	8220-4, 8220-8, 8220-16, 8220-32, 8230-8, 8230-16, 8230-32, 8260-8, 8260-16, 8260-32
	VS 8400	8420-8, 8420-16, 8420-32, 8430-8, 8430-16, 8430-32, 8460-8, 8460-16, 8460-32, 8460-8T[1], 8460-16T[1], 8460-32T[1], 8470-16, 8470-32, 8470-16T[1], 8470-32T[1], 8480-16, 8480-32, 8480-16T[1], 8480-32T[1]
CP10	*VS 10000*	10000/50-16, 10000/50-32, 10000/75-32, VS 10000/100-32
CP9	*VS 5000*	5430-1, 5430-2, 5440-2, 5440-4, 5440-8, 5450-2, 5450-4, 5450-8, 5460-4, 5460-8, 5460-16, 5660V-4T[1], 5660V-8T[1], 5660V-16T[1]
CP8	*VS 7000*	VS300-4, VS300-8, VS300-12, VS300-16, 7010-4, 7010-8, 7010-16, 7010-32, 7110-4, 7110-8, 7110-16, 7110-32, 7120-4, 7120-8, 7120-16, 7120-32, 7150-4, 7150-8, 7150-16, 7150-32, 7150-4T[1], 7150-8T[1], 7150-16T[1], 7150-32T[1] 7310-8, 7310-16, 7310-32, 7310-8T[1], 7310-16T[1], 7310-32T[1] All Above Models with 7011
CP7	VS 6	VS6-1AM[2], VS6-1BM[2],

		VS6-2AM[2], VS6-2BM[2], VS6-4AM[2], VS6-4BM[2]
	VS 6E	VS6E-1CM, VS6E-1DM, VS6E-2CM, VS6E-2DM, VS6E-4CM, VS6E-4DM
	VS 6T	VS6-1T[1], VS6-2T[1], VS6-4T[1], VS6-1CTT[1], VS6-1DTT[1], VS6-2DTT[1], VS6-4DTT[1]
	VS 65	VS65-1[2], VS65-2[2], VS65-4[2]
	VS 65T	VS65-2T[21], VS65-4T[21]
	VS 75E	VS75E-2DM, VS75E-4DM, VS75E-8DM
	VS 75E-T	VS75E-2T[1], VS75E-4T[1], VS75E-8T[1], VS75E-2DT[1], VS75E-4DT[1], VS75E-8DT[1]
CP4	VS 85	VS85-2C[2], VS85-4C[2], VS85-8C[2]
	VS 85T	7585VST-1[21], 7585VST-2[21], 7585VST-3[21]
	VS 100	VS100-64G[2], VS100-128G[2], VS100-256G[2], VS100-12[2], VS100-16[2]

A.2 CPU Controllers

The following list describes all of the CPU Controllers (by applicable CP-type, CP-series or -model, model number, and description) included in the C2 evaluated configuration.

A.2.1 Disk Controllers

CP Type	VS Series/Model	Controller Model Number
CP12	VS 8200 Series	23V98-4A, 70V98-4A
	VS 8400 Series	23V98-4, 23V98-4T[1], 23V98-4T[1], 70V98-4
CP10	VS 10000 Series	23V98-4, 70V98-4

[1]TEMPEST products are the same architecture as used in the commercial versions. They do NOT require any relevant modifications for the TCB or the C2 configuration tables. *tempesting* of VS Systems is primarily achieved by source suppression, and in some cases, source containment.

[2]These VS products are discontinued on current price lists; however, they will be continued as supported customer installed products until further notice.

CP9	VS 5000 Series[3]	50V68, 50V68-T[1] 50V98, 50V98-T[1]
CP8	VS 300 Series	23V98-4, 70V98-4
	VS 7010 Series	23V98-4A, 70V98-4A
	VS 7100 Series	23V98-4, 23V98-4T[1], 70V98-4
	VS 7300 Series	23V98-4, 23V98-4T[1], 70V98-4
CP7	VS 6/6E	25V50-2B, 25V50-4B
	VS 65/75E	25V50-2A
	VS 75E	25V50-4A, 25V98-4
	VS 6T/65T/75E-T	25V50-4AT[1]
	VS 6T/75E-T	25V98-4T[1]
CP4	All Models	22V28, 22V28A, 22V48
	VS 85-T	75V03T[1]

A.2.2 Magnetic Tape Controllers (MTCs)

This list only reflects available Magnetic Tape Controllers for selected CPUs that support parallel tape drive configurations.

CP Type	VS Series/Model	Controller Model Number
CP12	VS 8200 Series	23V95-1A, 23V95-2A
	VS 8400 Series	23V95-1, 23V95-2, 23V95-2T[1]
CP10	VS 10000 Series	23V95-1, 23V95-2
CP9	VS 5000 Series[4]	50V68, 50V68-T[1]
CP8	VS 300 Series	23V95-1, 23V95-2
	VS 7010 Series	23V95-1A, 23V95-2A
	VS 7100 Series	23V95-1, 23V95-2, 23V95-2T[1]
	VS 7300 Series	23V95-1, 23V95-2, 23V95-2T[1]
CP4	All Models	22V25, 22V15-2

[3]The VS 5000 Series also includes an embedded Resource Control Unit (RCU) that provides SCSI Controller support for up to seven SCSI compatible disk and tape devices. The RCU also provides support for the embedded 1.2 MB floppy diskette drive.

[4]Note: The VS 5000 Series also includes an embedded Resource Control Unit (RCU) that provides SCSI Controller support for up to seven SCSI compatible disk and tape devices.

A.2.3 Serial Device Controllers (SIOCs)

Some Serial IOCs have a modular design which allows it to be used in a variety of connection types. For example, the 23V67 SIOC can simultaneously support the MUXBUS and WANGNET channels. Up to sixty-four (64) serial devices may be logically connected. The conventional serial connection method is considered to be the "928" dual-coax cable via an Electrical Active Port Assembly (EAPA) back-panels for all controller models. Other serial device connections depending on the controller model and the attached Active Port Assembly (APA) may also be: Fiberway I & II fiber optics (see FW/FWII section), Twisted Pair APA via Balun, and WANGNET Peripheral Band Service (see WANGNET/P-Band section).

CP Type	VS Series/Model	Controller Model Number
CP12	All Models	23V67, 23V67-T[1]
CP10	All Models	23V67
CP9	VS 5000 Series	50V67, 50V97W, 50V97W-T[1]
CP8	All Models	23V67, 23V67T[1]
CP7	All Models	25V67
	VS 65T	25V67T[1]
CP4	All Models	22V27
	All Models	22V47
	VS 85T	75V02T[1]

A.2.4 Asynchronous Device Controllers (ADCs)

Unless otherwise specified as eight (8) port configurations, ADC controllers utilize two half-panel TC-APAs with eight (8) RS-232C connectors per panel. Each ADC IOC supports up to sixteen (16) devices which include 2110A Async Workstations, 2210 Async Terminal Emulation, ANSI X3.64 Async Terminals, and Wang Async Printers.

CP Type	VS Series/Model	Controller Model Number
CP12	VS 8200 Series	23V86-16A, 23V86A-ASYNCH
	VS 8400 Series	23V86-16A, 23V86A-ASYNCH
CP10	VS 10000 Series	23V86-16D, 23V86D-ASYNCH
CP9	VS 5000 Series[5]	50V96
CP8	VS 300 Series	23V86-ASYNCH
	VS 7010 Series	23V86-16A, 23V86A-ASYNCH

	VS 7100 Series	23V86-16, 23V86-ASYNCH
	VS 7300 Series	23V86-16, 23V86-ASYNCH
CP7	VS 6/6E	25V36B
	VS 15/65/75E	25V36A
	All Models	25V36AE
CP4	All Models	22V36

A.2.5 Local Asynchronous Device Controllers (LADCs)

LADC controllers utilize two half-panel TC-APAs with four (4) connectors per panel. A Modular Adapter plugs into one of the four connectors on the half-panel. Each Modular Adapter provides eight (8) RJ-11 ports for single-user cables which connect to peripheral devices. Each LADC IOC supports up to sixty-four (64) devices which include 2110A Async Workstations, 2210 Async Terminal Emulation, ANSI X3.64 Async Terminals, and Wang Async Printers.

CP Type	VS Series/Model	Controller Model Number
CP12	VS 8200 Series	23V46-A
	VS 8400 Series	23V86-16A, 23V86A-ASYNCH
CP10	VS 10000 Series	23V46-D
CP8	VS 7010 Series	23V46-A
	VS 7100 Series	23V46
	VS 7300 Series	23V46

A.2.6 Wangnet / Peripheral Band Controllers (P-Band)

WANGNET is a broadband cable providing many channel allocations and purposes. P-Band occupies one channel (i.e., a specified range of frequencies) for the purpose of providing 928/COAX-like serial connectivity on broadband cable plants for workstations and printers. Each system requires a WANGNET/P-Band IOC while the workstations and printers are connected via a NETMUX. Logical/physical addressing is accomplished by mapping the peripheral device logical address via GENEDIT to a particular port on the NETMUX. Each NETMUX is physically addressed on the cable plant with its own unique hardcoded address (Manufacturer's ID) and continuously polled by its corresponding IOC. Each IOC may be configured to address multiple NETMUXes via GENEDIT using the NETMUX's hardcoded address in the IOC device table. Simply put, the peripheral's logical device address is physically mapped to the concatenation of the IOC address to the NETMUX address to the physical port on the NETMUX where the peripheral is attached.

[5]Note: The VS 5000 Series also includes an embedded Resource Control Unit (RCU) that provides Asynchronous Controller support for up to four RS-232C asynchronous or printers.

CP Type	VS Series/Model	Controller Model Number
CP12	VS 8000 Series	VS-WN-28B
CP10	VS 10000 Series	VS-WN-28B
CP8	VS 7000 Series	VS-WN-28B
CP7	All Models	VS-WN-28C
CP4	VS 85	22V67W-A-28
	VS 90/100	22V67W-28
All	All Models	NETMUX-28, TP-NETMUX

A.2.7 Fiberway I & II Fiber Optics (FW/FWII)

Fiberway provides a fiber optic replacement medium for 928/COAX cables for interconnecting workstations and printers to the VS host. The host system must have provide a FW APA back panel from the appropriate Serial IOC to convert the electrical signal to fiber optics. Peripherals are attached to Fiberway using Fiber Optic Converters and/or the 16 port Remote Cluster Switch for converting the light signals to electrical COAX connections.

CP Type	VS Series/Model	Controller Model Number
All	All Models	FW-APA-2S, FW-RCS-16S, FO-MC-1S, FO-MC-1SA, FWII-SYS-4S, FWII-RCS-16S, FWII-ACA-8S
All	TEMPEST Models	FW-APA-2T[1], FW-RCS-16T[1], FO-MC-1T[1], FO-MC-1TA[1]

A.3 Peripheral Devices

The following list describes all of the Peripheral Devices included in the C2 evaluated configuration.

A.3.1 Disk Drives

Disk Storage Devices - SMD/ESMD

Depending on the particular CP type, SMD and ESMD disk drives may be internally mounted within the CP cabinet, within a Data Storage Cabinet (DSC), or as with older model drives, within their own self-contained

cabinet. For the purpose of listing the evaluation peripherals, only the actual drive type model numbers are listed.

CP Type	VS Series/Model	Peripheral Model Number
CP12	VS 8000 Series	2265V-1, 2265V-2, 2265V-3, 2267V-1, 2268V-1, 2268V-2, 2268V-3, 2268V-4, 2268V-6D
	VS 8000T Series	7565V-2T[1], 2267V-1T[1], 2268V-4T[1]
CP10	VS 10000 Series	2265V-1, 2265V-2, 2265V-3, 2267V-1, 2268V-1, 2268V-2, 2268V-3, 2268V-4, 2268V-6D
CP9	VS 5000 Series	2267V-1, 2268V-2, 2268V-3, 2268V-4
	VS 5000T Series	2268V-4T[1]
CP8	VS 7000 Series	2265V-1, 2265V-2, 2265V-3, 2267V-1, 2268V-1, 2268V-2, 2268V-3, 2268V-4, 2268V-6D
	VS 7000T Series	7565V-2T[1], 2267V-1T[1], 2268V-4T[1]
CP7	All Models	2265V-1, 2265V-2, 2265V-3, 2280V-1, 2280V-2, 2280V-3
	TEMPEST Models	7565V-2T[1]
CP4	All Models	2265V-1, 2265V-2, 2265V-3, 2280V-1, 2280V-2, 2280V-3
	VS 85T	7565V-2T[1]

Disk Storage Devices - SCSI

Depending on the particular CP type,, SCSI disk drives may be internally mounted within the CP cabinet and/or within a, SCSI Storage Module (SSM) or a, SCSI Mini Data Storage Cabinet (MDSC). For the purpose of listing the evaluation peripherals, only the actual drive unit type model numbers are listed.

CP Type	VS Series/Model	Peripheral Model Number
CP9	VS 5000 Series	2269V-3, 2269V-4, 2269V-5
	VS 5000T Series	2269V-3T[1], 2269V-4T[1], 2269V-5T[1]
CP7	VS 6	2269VR-3B, 2269VR-4B
	VS 6E	2269V-3B, 2269V-4B
	VS 6T	2269V-3T[1], 2269V-4T[1], 2269V-5T[1]
	VS 75E	2269V-4A

VS 75E-T	2269V-3T[1], 2269V-4T[1], 2269V-5T[1]

Disk Switch Options

Disk Switches allow both A and B cabling of a disk drive as non-concurrent dual channel connected devices. Only one system in a dual-channel connected configuration may logically attach a drive unit at any given time. This requires an operator to logically detach the drive unit from one system before logically attaching the drive unit to the second system. Upon an IPL, the first system to access the drive will retain logical possession of the drive unit until further operator intervention.

CP Type	VS Series/Model	Peripheral Model Number
All	Drive Dependent	SW04, SW04-1, SW04-1T[1], SW04-2, SW04-3, SW04-4, SW04-5, SW04-6, SW04-7

A.3.2 Tape Drives

Magnetic Tape Storage Devices - Parallel

CP Type	VS Series/Model	Peripheral Model Number
CP12	VS 8000 Series	2209V, 2209V-2, 2219V-1, 2219V-3, 2248V-1
	VS 8000T Series	2248V-1T[1]
CP10	VS 10000 Series	2209V, 2209V-2, 2219V-1, 2219V-3, 2248V-1
CP9	VS 5000 Series	2238V-1
CP8	VS 7000 Series	2209V, 2209V-2, 2219V-1, 2219V-3, 2248V-1
	VS 7000T Series	2248V-1T[1]
CP4	All Models	2209V, 2209V-2, 2219V-1, 2248V-1
	VS 85-T	2248V-1T[1]

Magnetic Storage Devices - Serial

CP Type	VS Series/Model	Peripheral Model Number
All	All Models	2238V-1, 2509V, 2529V
	All TEMPEST	2238V-1T[1], 7529T[1]

Magnetic Tape Storage Devices - SCSI

CP Type	VS Series/Model	Peripheral Model Number
CP9	VS 5000 Series	2238V-3H

A.3.3 Workstations

Workstations

CP Type	VS Series/Model	Peripheral Model Number
CP9	VS 5000 Series	2110A, 2246C, 2246K, 2246S, 2246SI, 2256C, 2256MWS, 2266C, 2266S 4210UK, 4210BK, 4210WM, 4210NL 4230, 4230A, 4230AL, 4230MWS, 4245, 4245MWS, 6300GM 4430, 4430MWS[6]
All other CPs	All Models	2246S, 2246C, 2246K, 2246SI, 2256C, 2256MWS, 2266S, 2266C, 2276C, 4210WM, 4210NL, 4210UK, 4210BK, 4230, 4230A, 4230AL, 4230MWS, 4245, 4245MWS, 4430, 4430MWS[6] 6300GM
All	All TEMPEST	4230-VS-T[1], 7501-VS-T[1]

A.3.4 Printers

Printers - Band

CP Type	VS Series/Model	Peripheral Model Number
All	All Models	5573, 5574, 5574-1, 5575, 5575X
	All TEMPEST	5574-1T[1]

[6] The 4430 workstations are only supported on the VS 5000 Series, VS 7000 Series, VS 8000 Series, and VS 10000 Series.

Printers - Matrix

CP Type	VS Series/Model	Peripheral Model Number
All	All Models	5577, 5578, LM-400, LM-700, LM-900

Printers - Laser

CP Type	VS Series/Model	Peripheral Model Number
All	All Models	LCS15-2, LDP8, LIS12, LIS24, LPS8, LPS12
	All TEMPEST	LCS15-T[1], LDP8-T[1], 75LIS-12VT[1]

Printers - Daisy

CP Type	VS Series/Model	Peripheral Model Number
All	All Models	DWOS55, DWOS60
	All TEMPEST	75DW/OS-55T[1], DW/OS-60T[1]

Printers - Asynchronous

CP Type	VS Series/Model	Peripheral Model Number
All	All Models	DM50/300, PM015, PM018, PM017, PM019

Appendix B

Evaluated Software Components

SVS/OS PACKAGES (TCB[1])	Version
VS Operating System Package (VS/OS)	7.33.36
Device Support Package (DSP)	4.90.20
WP Printer Task Procedure (WPPRT)	
WP Printer Task Object Module (WPTCB)	
Multiwindow (MWS) Microcode - @2256MWS[2]	
Multiwindow (MWS) Microcode - @4230MWS[2]	
Multiwindow (MWS) Microcode - @4245MWS[2]	

CPU MICROCODE (TCB[1])	Version
CP4 Microcode	4.70.46
CP7 Microcode[3]	7.70.27
CP7 Microcode w/ Floating Point[3]	7.71.27
CP8 Microcode	8.70.63
CP8 Microcode w/ Floating Point	8.71.63
CP9 Microcode[3]	9.70.17
CP9 Microcode w/ Floating Point[3]	9.71.17
CP10 Microcode	10.70.07
CP12 Microcode	12.70.06

SCU PACKAGE (TCB[1])	Version
Support Control Unit (SCU, CP8/10/12 Only)	1.99.00

O/S INSTALLATION PACKAGES (TCB[1])	Version
SAU / VS 65 (Standalone Utilities)	1.03.15
SAU / VS 75E (Standalone Utilities)	1.03.15
OSINSTALL / CP 8 (O/S Installation Utility)	00.00.06
OSINSTALL / CP 9 (O/S Installation Utility)	00.00.06
OSINSTALL / CP 10 (O/S Installation Utility)	00.00.06
OSINSTALL / CP 12 (O/S Installation Utility)	00.00.06

[1] These modules will be auto-enclosed in the VS/OS or DSP software packages as they are TCB relevant to the release.

[2] These modules are not auto-enclosed in the VS/OS or DSP software packages.

[3] An asterisk (*) denotes these CP microcode files are resident in their respective diagnostic IPL packages. The diagnostic package is required in order to properly boot and IPL the system with the CP microcode. This is applicable to the Bus Processor and RCU based systems i.e., CP 7 and CP 9 respectively.

DIAGNOSTIC PACKAGES (Non-TCB)	Version
CP4 / VS 100 Micro Diagnostics Package	Multi-Revs
CP7 / BP1 Diagnostic Package (VS 65)	2.00.00
CP7 / BP2 Diagnostic Package (VS 6/6E/75E)	2.00.00
CP8 Diagnostic Package	REV 2990
CP9 / VS 5000 IPL & Diagnostic Package	2.02.00
CP10 Diagnostic Package	REV 2990
CP12 Diagnostic Package	REV 2990
VS Online Tests & Monitor Diagnostic Package	REV 2951
VS Online FTU Package	REV 6935

SUPPORTING O/S SERVICES PACKAGES (Non-TCB)	Version
FMU (File Management Utilities)	2.98.03
VS SUBS (Program Developers' Subroutines)	1.00.07

Appendix C

Glossary

Access Control List (ACL) A list of users and/or groups and their access (write, read, execute or null) used to control access to a file.

Access Mode One of four hierarchical representations (W — write; R — read; E — execute; blank — no access) used to specify the way a user can access protected files.

Alphanumeric A character set that contains letters, digits and other characters, such as punctuation marks. Wang's standard set of such characters is A-Z, 0-9, and #, @, and $.

Arithmetic and Logic Unit (ALU) That portion of the CPU which performs mathematical calculations and data comparisons.

Background Processing A workstation-independent, noninteractive processing environment that consists of one or more programs and procedures run by a controlling procedure.

Basic Assembly Language (BAL) The assembly language for Wang VS systems.

Bus Adapter (BA) An interface device that provides a mechanism for a CP to communicate with one or more IOPs through the system's I/O bus.

Bus Interface Controller (BIC) An interface device which allows an I/O coprocessor to directly communicate with the system bus.

Bus Processor (BP) A device which performs as an IOC that communicates with a device adaptor (DA).

Central Processor (CP) The central processing unit (CPU). That component of the computer composed of electrical circuitry which directs most of the computer's activities, and which consists of the control unit, the arithmetic/logic unit (ALU), and the processing registers.

Child Task A task that has been created by another task through the `TINVOKE` system service.

CP Type A set of CPs that are based on the same hardware architecture (i.e., *CP5*, *CP8*, etc.).

Command Table Address (CTA) An address found within the I/O status table (IOST). A specific CTA entry points to the I/O command table (IOCT) for the IOC associated with a given I/O operation.

Control Mode A state of the computer system in which normal program execution is halted and special facilities for diagnostics, restart, initialization, and debugging are made available.

Data Management System (DMS) The operating system routines that control data file creation and access methods at the record, as opposed to file, level. DMS performs all input/output operations and controls the physical location of information on each volume.

Datagram A message sent from one task to another's mailbox. Any task can send a datagram to a mailbox for which datagrams are allowed.

Dedicated System Task A privileged task not identified with a logged on user, but with the operating system, instead.

Device Adapter (DA) An interface device that provides the functionality to talk with a given device. The DA, in conjunction with a bus processor (BP), handles all communication between the CP and devices.

Diagnostician A user that has been given the diagnostic privilege, which allows a task to load microcode to an IOC, and gives it the ability to read and write IOC data structures. This privilege is designed to allow specified users to run on-line diagnostics via system service routines.

Direct Memory Access (DMA) A method, used by controllers, to access main memory without CPU intervention.

Double-Word 64 bits.

Error Correction Circuitry (ECC) That feature of hardware which detects and corrects single bit errors and which detects and reports double bit errors but does not correct them.

Evaluated Products List (EPL) A list of evaluated products, the aim of which is to provide ADP system developers, managers, and users an authoritative evaluation of a system's relative suitability for use in processing sensitive information. The EPL is maintained by the NCSC.

Extent A group of physically contiguous records of a disk allocated for use by a single file.

File A logical unit consisting of zero or more records of related information.

File Protection Class One of a set of classes used to control file access. A file protection class is designated by a one-character file class protection code. There are 30 disjoint file classes which can be divided into two types, noted as Type I and Type II.

File protection class access rights A user's access to each of the 26 (A-Z) type II file protection classes.

Foreground Processing An interactive process associated with a workstation.

Group A list of user IDs which the SA defines and maintains, and which is used in access control lists. Each group is identified by a group ID.

Half-Word 16 bits.

Initial Program Load (IPL) The process of loading information into the system that it needs to start running.

Initiator task One of the tasks started by the System Operator as background tasks to which work can be assigned by users with the `SUBMIT` service.

I/O Command Table (IOCT) A table of I/O command words (IOCWs) for the devices attached to an I/O processor. A command table address (CTA) specifies the address in main memory of each IOCT.

I/O Command Word (IOCW) A variable-length (1 to 8 bytes) that specifies the next command to be executed for a device. Byte 0 holds the command code. Bytes 1-3 hold a data address or beginning address of an indirect address list. Bytes 4-5 hold the data count (number of bytes to be operated on or transferred), and bytes 6 to the end may hold additional device-dependent information.

I/O Controller (IOC) An interface between the CP and an I/O device.

I/O Processor (IOP) The processor that controls the transfer of data between main storage and peripheral units, relieving the CPU of this slower function.

I/O Status Table (IOST) A table which contains information about completed I/O operations. There is one 16-byte entry in the IOST for each supported controller on the system.

I/O Status Word (IOSW) Eight bytes of data, stored at main memory location 0, that pass information concerning the status of an I/O device to the CPU. Byte 0 is the general status byte. Byte 1 is the error status byte, stored if the error completion bit is set to 1 in the general status byte. Bytes 2-3 are the device-dependent bytes; bytes 4-5 hold the residual byte count. Bytes 2-6 (bits 16-55) are sometimes referred to as the extended status bits for disk and tape.

Job A background task submitted for processing using the `SUBMIT` command or SVC.

Jump to Subroutine on Condition Indirect (JSCI) A machine instruction used to transfer program control to a linked subroutine.

Kilobyte (KB) 1024 (2^{10}) bytes.

Library A collection of files, often of related contents, identified by a library name.

Link Level A level associated with a program. Each link level is represented by a data structure called the program file block, and a save area. Any files or program libraries introduced into the address space in a link level are deleted when the subtask completes and returns to its caller.

Local Page Table (LPT) In local memory a table consisting of one byte for each page in a segment. This byte contains the physical page number of that page when it is in main memory. There is one local page table for each region of virtual memory. These tables are used by the CPU in translating virtual memory addresses to main memory addresses.

Logon ID A unique character string (1 to 8 characters in length) which the SA assigns to a user. The logon ID is entered by the user at logon time.

Machine Language Architecture (MLA) The interface presented by the microcode on the various CP types. This architecture is the same across all CPs and defines such things as instructions, process levels, etc.

Mailbox A system memory data structure used in interprocess communication. It is used with datagrams and with sessions.

Megabyte (MB) 2^{20} bytes.

Modifiable Data Area See "User Modifiable Data Area".

Nucleus That part of the software, maintained in the file @SYS00n@, (where n represents the CP type) that must be resident, or is called by a routine that is resident.

Operator A specific role assigned to a user by the SA. In this role a logged on user can perform specific system functions. The SA specifies this role by giving a user any or all of the operator functions (Resource Access Privileges) by using the VSSECURE utility.

Parent Task A task that invokes another task by issuing the TINVOKE system call.

Personal Computer (PC) A microcomputer usually intended for the use of a single user.

Port A one-way communication path that is implemented in system-maintained data structures that supports messages of up to 2046 bytes in length. Ports are named by the creator with a system unique 4-character name.

Print Class A set of 26 classes (A-Z) to which print files are assigned. Each printer is assigned certain print classes, so a file's print class determines which printers can print it. These are used for special forms handling and scheduling.

Procedure File A file containing system commands, user parameters, and statements unique to the procedure language which is used to control the execution of a task.

Process See "task".

Process Level One of eight levels (0 - 7) assigned to the current execution state of the processor. This level determines the degree of privilege assigned to the current task. Virtual memory accessible to a given level is accessible to all higher process-level tasks, as well.

Program Control Word (PCW) An eight-byte code that contains program and system status information and which enables communication to take place between the program and the operating system.

Programmable Read-only Memory (PROM) Non-volatile memory which functions in a manner similar to read-only memory (ROM), except the data or program instructions are not pre-recorded on the PROM chip when it is manufactured, thus allowing users this capability.

Page Frame 2048 contiguous bytes, starting on a 2048 byte boundary.

Page Table A section of main memory that defines the mapping of virtual address space to physical address space.

Random Access Memory (RAM) Volatile temporary storage for data and program instructions.

Record A collection of bytes up to 2048 bytes (2 KB) in length.

Reference and Change Table (RCT) A table which associates two bits — the reference and the change bit — with each page frame.

Region A part of the user's virtual memory space consisting of a varying number of contiguous pages.

Region Node A 16-byte entry in the Region Table that describes a particular region and points to the page table for that region.

Region Node Table (RNT) A section of main memory that describes the user's virtual address space. It contains one region node entry for each region defined for the user.

Resource Control Unit (RCU) A special I/O coprocessor which, for CP9 systems, allows Workstation 0 to interface with the system bus. The RCU orchestrates the IPL process.

Role One of five categories to which tasks are assigned. Four of these are for assignment to users.

Segment Control Register (SCR) A privileged 32-bit register holding the address of a task's main memory page table for a segment.

Semaphore A 2-word data structure that is used to synchronize processes.

Session A logical two-way communication between tasks that have agreed to communicate, with the mailboxes on either end acting as gateways. Once agreement is reached, each task is free to send and receive messages on the session. A mailbox is used at each end of the session as the repository for the messages.

Special Program Access Rights (SPAR) Rights comparable to file protection class access rights, but associated with a program, not a user. They can only be specified by the SA. They specify that the program has SA access rights (i.e., it executes with the role of the SA), or the access that the program has to each of the 26 type II file protection classes. A dynamically linked program receives the task's access rights as well as its own SPARs. In addition, a calling task can specify whether a linked program also receives an accumulation of all the SPARs of the intermediately linked programs.

Stack Header Block (SHB) A data block describing the stack associated with one of a task's process levels. Entries in an SHB hold the stack vector and the address of the most recently-built JSCI save area. SHBs are referred to by the operating system to control stack switching.

Stack Header Block Table (SHBT) A table of eight addresses, each pointing to the stack header block associated with one of a task's process levels.

Supervisor Call (SVC) A machine instruction used by a task to invoke an operating system service.

Support Control Unit (SCU) A 16-bit microprocessor-based unit with a built-in 5.25 inch diskette drive and a fixed-disk drive. The SCU consists of the system panel and the workstation that is cabled as device 0 on port 0, and, for CP8, CP10, and CP12 processors is the system console. The SCU runs a special applications software package known as SYSCON.

System Administrator (SA) The SA role is a specified account that is shipped with the system, with a predefined password. The SA can specify other users as an SA by using the **VSSECURE** utility to set an indicator in the user profile. An SA has autonomous control over the entire system.

System Bus Interface (SBI) An interface between the system bus and the I/O bus, thus allowing these buses to communicate.

System Console The system console consists of the system panel and that workstation attached as device 0 on port 0. On VS7000 Series processors the System Console is the SCU.

System Services Those operating system functions called by user and system applications to perform file management, I/O, memory management, and other services involving system resources. These services are available to programs through either the **JSCI** or **SVC** machine instructions.

Task An operating system schedulable entity to which system resources are allocated.

Translation Buffer (T-BUF) The local page table of the VS7310, VS10000/100, and VS8400.

Translation RAM (T-RAM) A local page table (i.e., an area of local CP memory holding page frame numbers for the loaded portion of a task's virtual address space. This special hardware memory is separate from the main memory of the computer).

Trusted Computing Base (TCB) The totality of protection mechanisms within a computer system – including hardware, firmware, and software – the combination of which is responsible for enforcing a security policy. A TCB consists of one or more components that together enforce a unified security policy over a product or system. The ability of a TCB to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters (e.g., a user's clearance) related to the security policy.

Unit Control Block Address (UCBA) The address of a unit control block (UCB) for a device. A UCBA is stored within an I/O command table.

Unprivileged User Any user that is not an operator, a diagnostician or a system administrator.

User ID A three-character unique code, assigned by the SA, which identifies the user to the system.

User Lock A data structure available to all user tasks to coordinate access to shared resources. This mechanism is primarily available to coordinate access to shared files that have been mapped, by the **MSMAP** system service, to an area of the task's virtual address space.

User Modifiable Data Area A portion of a task's virtual memory used by a task running at process level 0 to store program variables and buffers, as well as the stack. The size of the user modifiable data area is fixed for each user by the SA.

Virtual Address A 24-bit address consisting of a 13-bit virtual page index and an 11-bit byte offset.

Volume A portion of a single unit of storage that is accessible to a single read/write mechanism. A disk pack and a floppy disk are examples of volumes.

Volume Protection Label (VPL) An adhesive paper label that the operator must physically attach to each tape volume.

Volume Set A group of one or more volumes used to store very large files or very many related files.

Wait level User tasks are assigned to one of two wait levels. Wait level 1 is the user wait level, and wait level 2 is the command processor/operator interface wait level. The system maintains separate areas in control blocks for a task's context (including screen contents) for each of these two wait levels.

Word 32 bits.

Bibliography

[1] *SVS/OS with Controlled Access Protection (CAP) RM-Plan*, Wang Laboratories, Inc., 1990.

[2] *Rating Maintenance Phase Program Document*, National Computer Security Center, 23 June 1989, NCSC-TG-013. June 1989.

[3] *VS Trusted Facility Manual (TFM) Documentation Guide*, Wang Laboratories, Inc., 1990, 715-3692.

[4] *VS5000 Series Processor Handbook*, Wang Laboratories, Inc., October 1988.

VS7000-T Series Processor Handbook, Wang Laboratories, Inc., May 1988.

VS7100 Series Processor Handbook, Wang Laboratories, Inc., July 1987.

VS7810 Processor Handbook, Wang Laboratories, Inc., August 1987.

VS8200 Family Processor Handbook, Wang Laboratories, Inc., September 1989.

VS10000 Series Processor Handbook, Wang Laboratories, Inc., February 1989.

[5] *Secure VS Operating System with Controlled Access Protection Guide*, Wang Laboratories, Inc., 1990, 715-2749.

[6] *Secure VS Operating System with Controlled Access Protection (SVS/OS with CAP) Release 1 Software Bulletin* Wang Laboratories, Inc., June 1990.

[7] *Secure VS Operating System with Controlled Access Protection (SVS/OS with CAP) Release 1 Customer Software Release Notice (draft)*, Wang Laboratories, Inc., September 1989.

[8] *VS GENEDIT Utility Reference*, Release 7 Series, Wang Laboratories, Inc., December 1989.

[9] *VS BACKUP and VOLCOPY User's Guide*, Wang Laboratories, Inc., 1990, 715-2383.

[10] *Department of Defense Trusted Computer System Evaluation Criteria*, December 1985, DOD 5200.28-STD.

[11] *VS Data Management System Reference*, 1st Edition, Wang Laboratories, Inc., 1984, 800-1124-01.

[12] *VS DMS/TX Reference*, 1st Edition, Wang Laboratories, Inc., 1983, 800-1128-01.

[13] *VS Extended Data Management System (XDMS) Reference*, 1st Edition, Wang Laboratories, Inc., 1988, 715-1159.

[14] *VS Operating System and System Utilities Documentation Guide*, Release 7 Series, Wang Laboratories, Inc., 1988, 715-0296A.

[15] *VS Procedure Language Reference*, 6th Edition, Wang Laboratories, Inc., 1987, 800-1205-06.

[16] *VS Assembly Language Reference*, 3rd Edition, Wang Laboratories, Inc., 1982, 800-1200AS-03.

[17] *VS Operating System and System Utilities Error Message Reference*, 1st Edition, Wang Laboratories, Inc., 1988, 715-0621.

[18] *VS System Security Test Reference, C2 Certification Test Documentation*, Volume 1-9, Version 1.00.00, Wang Laboratories, Inc., 1988, 714-0193 (proprietary).

[19] *VS File Management and Application Development Utilities Reference*, 1st Edition, Wang Laboratories, Inc., 1988, 715-1715.

[20] *VS System Media, Transfer, and Device Utilities Reference*, 1st Edition, Wang Laboratories, Inc., 1988, 715-1716.

[21] *VS System Administration and Analysis Utilities Reference*, 1st Edition, Wang Laboratories, Inc., 1988, 715-1717.

[22] *VS System User's Introduction*, Release 7 Series, Fourth Edition, Wang Laboratories, Inc., 1988, 715-0417C.

[23] *VS Editor Reference*, Release 7 Series, 1st Edition, Wang Laboratories, Inc., 1988, 715-1143.

[24] *VS Linker Reference*, Release 7 Series, 1st Edition, Wang Laboratories, Inc., 1987, 715-1145.

[25] *VS Symbolic Debugger Reference*, Release 7 Series, 1st Edition, Wang Laboratories, Inc., 1987, 715-1144.

[26] *VS VSSUBS Reference*, 1st Edition, Wang Laboratories, Inc., 1988, 715-0599.

[27] *VS Principles of Operation, Volumes I and II*, Wang Laboratories, Inc., 1990. Volume 1 (715-4098) is the customer version. Volume 2 (INT-0495) is proprietary.

[28] *VS System TCB CPU Architecture Analysis, VS/OS 7.30 TCB/C2*, Version 1.0, Wang Laboratories, Inc., April 1988, (proprietary).

[29] *Potential CPU/Peripheral List for TCB/C2.*

[30] *VS System Informal Security Policy Statement*, Version 2.1, Wang Laboratories, Inc., 1988, INT-0496 (proprietary).

[31] *VS System Informal Security Policy Statement: Wang Response to NCSC Questions, VS/OS 7.30 TCB/C2*, Wang Laboratories, Inc., March 1988 (proprietary).

[32] *VS/OS TCB Definition and Mapping Document*, Wang Laboratories, Inc., March 1988 (proprietary).

[33] *VS Security Features User's Guide*, NCSC Draft, Wang Laboratories, Inc., 1988, 715-0793.

[34] *VS Operating System Services Reference*, Volumes 1 and 2, Wang Laboratories, Inc., 1990, 715-2344, 715-2345.

[35] *Trusted Computing System Installation and Administration Guide*, NCSC Draft, Wang Laboratories, Inc., April 1988.

[36] *VS Enhanced System Access Controls (ESAC) Guide*, NCSC Draft, Wang Laboratories, Inc., 1990, 715-4037.

[37] *VS GENEDIT Utility Reference*, Release 7 Series, 1st Edition, Wang Laboratories, Inc., 1988, 715-1511.

[38] *VS System Operator's Guide*, Wang Laboratories, Inc., 1990, 715-3536.

[39] *VS Operating System Internals Manual*, 1st Edition, Wang Laboratories, Inc., 1990, INT-0492 (proprietary).

[40] *VS Utilities Internals Manual*, 1st Edition, Wang Laboratories, Inc., 1990, INT-0494 (proprietary). 715-1839.

[41] *VS System Development Tools* , Version 5, Wang Laboratories, Inc., 1990, INT-0493 (proprietary). 714-0195.

[42] *WP Queue Entries in the OS/VS Queue, Project Proposal*, Version 1.2, 28 September 1987 (proprietary).

[43] *SUBMITting WP Print Requests, Functional Specification*, Version 1.1, 23 October 1987 (proprietary).

[44] *SUBMITting WP Print Requests, Design Specification*, Version 1.5 , 31 March 1988 (proprietary).

[45] *Integrated WP/DP Printing-WP Component, Functional Specification*, 28 July 1987 (proprietary).

[46] *Group Editor Using PACE, Functional Specification*, Version 3.4, 20 January 1988 (proprietary).

[47] *Group Editor Using PACE, Design Specification*, Version 1.3, 16 February 1988 (proprietary).

[48] *Prevent GROUPLST RE-USE, Functional Specification*, Version 2.1, January 15, 1988 (proprietary).

[49] *Fault Tolerant Event Logging, Functional/Design Specification*, Version 3.0, 29 February 1988 (proprietary).

[50] *C2 Disk Mount Procedures*, 30 March 1988 (proprietary).

[51] *IPC/XMIT Message Auditing Facility, Functional Specification*, Version 1.2, November 17, 1987 (proprietary).

[52] *IPC/XMIT Message Auditing Facility, Design Specification*, Version 1.2, November 17, 1987 (proprietary).

[53] *VS Operating System, Release 7.18, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1988, 715-2208.

[54] *VS Device Support, Release 2.99, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1988, 715-2235.

[55] *VS 5000 Series Diagnostics, Release 1, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1988, 715-2162.

[56] *VS VSSUBS, Release 1.00.00, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1988, 715-2233.

[57] *VS File Management and Application Development Utilities, Release 00.00.07, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1988, 715-2234.

[58] *VS Standalone Utilities (SAU), Release 1.03.13, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1988, 715-2232.

[59] *VS Support Control Unit (SCU), Release 1.05, Customer Software Release Notice*, 1st Edition, Wang Laboratories, Inc., 1987, 715-1508.

[60] *VS7100 Series Processor Handbook*, Wang Laboratories, Inc., July 1987.

[61] *Subsets of Standard Code for Information Interchange, Federal Information Processing Standards Publication 15*, U.S. Department of Commerce, National Bureau of Standards, 1 October, 1971.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT UNLIMITED DISTRIBUTION
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CSC-EPL-90/004	5. MONITORING ORGANIZATION REPORT NUMBER(S) S236,000
6a. NAME OF PERFORMING ORGANIZATION National Computer Security Center 6b. OFFICE SYMBOL (If applicable) C71	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) 9800 Savage Road Ft. George G. Meade, MD 20755-6000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Final Evaluation Report Wang Lab Inc., SVS/OS CAP 1.0	

12. PERSONAL AUTHOR(S)
Timothy J. Bergendahl, Duane A. Souder, Anthony J. Apted, James L. Arnold Jr., Ronald J. Bottomly, R. Kris Britton

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr/ Mo/ Day)	15. PAGE COUNT
Final	FROM _____ TO ___	90,09,28	97

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	NSA, Wang Laboratories Incorporated, TCSEC, C2, SVS/OS Cap 1.0

19. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The National Security Agency's (NSA) Trusted Product and Network Security Evaluations Division examined the security protection mechanisms provided by Wang Laboratories, Incorporated's SVS/OS CAP 1.0. It was evaluated against the *DoD Trusted Computer System Evaluation Criteria (TCSEC)* and the evaluation team determined that the system meets all criteria for the C2 level of trust.

This report documents the findings of the evaluation

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL PATRICIA L. MORENO	22b. TELEPHONE NUMBER (Include Area Code) (301)859-4458	8b. OFFICE SYMBOL C71